Universidad Zaragoza
1542

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Proyecto Fin de Carrera

Ingeniería Industrial

Automatización Industrial y Robótica

Modelado y simulación de un robot Robucar TT

Carlos Gracia Sola

Director: Luis Montano Gella

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Mayo 2013

Modelado y simulación de un robot Robucar TT
Resumen

En este proyecto se ha creado un modelo para el robot Robucar TT del Grupo de Robótica, Percepción y Tiempo Real y posteriormente se ha implementado en el simulador Gazebo.

El Robucar es un vehículo con tracción a las cuatro ruedas que puede moverse en tres modos distintos. El primero es el llamado modo coche, donde las ruedas traseras se fijan y giran únicamente las delanteras. En el segundo modo, de cuatro ruedas, todas ellas se mueven para permitir radios de giro menores. El último modo es el modo cangrejo, donde todas las ruedas giran en la misma dirección para permitir desplazamientos en diagonal.

Un simulador de movimiento de un robot sirve para la puesta a punto de numerosas aplicaciones en robótica. Debe reproducir los más fielmente posible el comportamiento cinemático y dinámico del robot, además de sus distintos modos de tracción. Este comportamiento debería estar incorporado en el simulador. Sin embargo, lo más habitual es que no lo esté. Por tanto, en este proyecto se ha planteado incorporar las características obtenidas de la experimentación con el robot real Robucar TT.

Para ello se han planteado una serie de pruebas con el robot real y se ha comparado su comportamiento con el del simulado utilizando las mismas órdenes. Ello ha permitido el ajuste de los parámetros en el simulador, modificando el modelo inicial para que se ajuste mejor a la realidad. Dado que es inevitable que exista un error, se crea un modelo para éste y se estima su matriz de covarianzas.

Otro de los objetivos del proyecto era el estudio del modelado geométrico del robot y de su entorno a partir de un sensor RGB-D como una cámara Kinect. Se han desarrollado distintos métodos para ello. Se presentan los resultados obtenidos y se analizan sus limitaciones de aplicabilidad.

ÍNDICE DE CONTENIDOS

CAPÍTULO 1

INTRODUCCIÓN

El principal objetivo de este proyecto es el desarrollo de un simulador realista para un robot Robucar TT. Para ello se trabaja sobre el simulador Gazebo. Se puede ver una imagen del robot en la figura 1.1. Adicionalmente, se estudia la posibilidad de generar modelos tridimensionales tanto del robot como de su entorno utilizando un sensor Kinect para su visualización en el simulador.

1.1. Motivación y objetivos

A la hora de crear nuevos programas para un robot, un simulador puede resultar ser una herramienta muy útil. Permite realizar pruebas con más rapidez, seguridad y comodidad que utilizando el robot real. Además, puede recrear escenarios cuya utilización en la realidad sería más compleja, inviable o directamente imposible.

Como contrapartida, una simulación se basa necesariamente en un modelo simplificado de la realidad, por lo que su comportamiento nunca será exactamente igual. Algunos factores no se tendrán en cuenta y los demás no serán perfectamente realistas, así que nunca hay que perder de vista que un simulador no es más que una aproximación. A menudo, se utiliza como un paso previo antes de las pruebas reales, permitiendo eliminar los fallos más importantes en un entorno seguro para posteriormente comprobar y ajustar el funcionamiento en la realidad.

Un simulador para robots cuyo uso se ha extendido mucho recientemente es Gazebo [1]. Permite introducir varios robots en un mapa, además de sus sensores y otros objetos. Genera información realista para esos sensores y cuenta con un motor de física para simular la dinámica de sólidos rígidos.

El control de los robots se realiza mediante ROS, *Robot Operating System* [2]. Es otro sistema cuyo uso se ha generalizado en los últimos años. Presenta la

Figura 1.1: Fotografía del robot.

ventaja de que también es el sistema con el que se controla al propio robot real. Si el simulador se diseña correctamente, es posible utilizar exactamente los mismos programas para ambos robots.

Actualmente no hay publicado ningún modelo para simular un robot con las características del Robucar del Grupo de Robótica de la Universidad de Zaragoza. Existen simuladores para otros robots como el PR2 [3] y el Turtlebot [4], pero ninguno de ellos tiene las características de éste. El principal objetivo del presente proyecto es crear un modelo realista para el Robucar e implementarlo en Gazebo.

Este modelo incluye la geometría, la cinemática y la dinámica del robot. Se busca que el robot simulado se mueva de una manera similar al real bajo las mismas órdenes. Se tendrá en cuenta las causas que producen ese movimiento mientras sean útiles para lograr ese objetivo. Sin embargo, en el caso de que no sea posible modelar la dinámica con la suficiente precisión, se considerarán alternativas para imitar sus efectos. Por supuesto, eso no quiere decir que se ignoren las restricciones dinámicas del robot, que pueden ser muy importantes para lograr recrear correctamente la cinemática.

También es importante facilitar la modificación del modelo, tanto para su posterior ajuste como para permitir la posibilidad de que sea utilizado para robots con características similares.

Por último, se estudia la posibilidad de crear un modelo CAD del robot y de su entorno mediante el uso de un sensor Kinect [5], en un formato compatible con el visualizador de Gazebo. La reconstrucción en tiempo real de un entorno tridimensional se trata en [6], incluyendo el renderizado de imágenes. [7] trata

sobre el modelado de entornos interiores mediante el uso de una cámara tipo Kinect, pero no llega a generar las superficies que serían necesarias para el modelo CAD. El objetivo es llegar a crear ese modelo a partir de la información de diversas capturas de la cámara.

1.2. Estructura de la memoria

En este capítulo introductorio se han explicado brevemente las razones para llevar a cabo este proyecto y qué se pretende conseguir. En el capítulo 2 se describe cómo es el robot, prestando especial atención a su cinemática y su dinámica, con el nivel de detalle suficiente como para modelarlo. La creación e implementación de ese modelo se explica en el capítulo 3. Incluye información sobre el *software* utilizado y un análisis de sensibilidad del simulador para saber qué parámetros son los más importantes. El capítulo 4 explica las pruebas realizadas con el robot real y cómo se han usado para ajustar el comportamiento de la simulación a la realidad. También incluye la estimación del error. Cómo se ha tratado el modelo visual mediante el sensor Kinect se expone en el capítulo 5. El capítulo 6 contiene las conclusiones obtenidas durante la ejecución del proyecto y presenta algunas ideas sobre posibles trabajos futuros.

Después del cuerpo principal de la memoria se añaden una serie de anexos para completarla. El anexo A muestra fotografías del robot. Los mensajes de ROS que utiliza el simulador se explican en el anexo B, con la idea de facilitar su uso. El anexo C contiene los resultados de las pruebas realizadas para el análisis de sensibilidad del simulador. El anexo D incluye una descripción de los nodos de ROS que se utilizan. La información obtenida para el ajuste del simulador se incluye en el anexo E. Una explicación de los parámetros que se pueden modificar se encuentra en el anexo F. En el anexo G se explica cómo se calcula la matriz de covarianzas usada para la estimación del error del simulador.

También se incluye un CD-ROM con varias animaciones del simulador. El anexo H contiene información sobre el contenido de este CD-ROM.

Por último, se añade un índice de figuras, otro de tablas y la bibliografía consultada. La referencias que aparecen en esta última se ordenan por orden de aparición.

CAPÍTULO 2

DESCRIPCIÓN DEL ROBOT

En este capítulo se describe al robot con el nivel de detalle suficiente para poder crear posteriormente el modelo para la simulación. Se presta especial atención a la cinemática en sus distintos modos de movimiento, para saber cómo deben actuar las ruedas en función de la velocidad requerida al robot.

2.1. El Robucar

El robot sobre el que trata este proyecto es un Robucar, propiedad del Grupo de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza y fabricado por la empresa francesa Robosoft. En concreto, es un Robucar TT modificado. Se le han añadido varios elementos como más sensores láser, un brazo robótico y una cámara térmica. La figura 2.1 muestra fotografías de un Robucar básico y del modificado para el grupo. El anexo A contiene más fotografías del robot.

Los principales parámetros que se van a considerar para aproximar en el simulador el comportamiento real del robot son:

- Geometría: dimensiones del robot y de las ruedas.
- Cinemática: modos de tracción o movimiento.
- Dinámica: fricción, masa total, distribución de masa.
- Límites de velocidades y aceleraciones.

(a) Robucar TT

(b) Configuración modificada

Figura 2.1: La fotografía (a) muestra un Robucar TT en su versión FAST mientras que la (b) muestra la configuración modificada del robot sobre el que trata el presente proyecto.

2.1.1. Posicionamiento

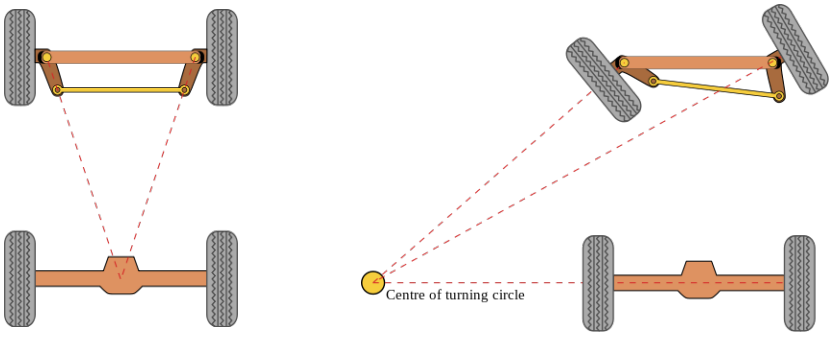
Para conocer su posición, el robot utiliza una IMU, *Inertial Measurement Unit* o Unidad de Medición Inercial. Ésta se compone de una serie de acelerómetros y giróscopos que permiten saber cuánto se mueve y gira el robot [8]. Es muy precisa, pero al basarse en el movimiento va acumulando error.

Las pruebas que se realizan en este proyecto son suficientemente cortas como para que no sea necesario tenerlo en cuenta. Para caracterizar el comportamiento del robot no hay que efectuar ninguna prueba más larga, dado que éstas se pueden separar en varias más cortas. Por ejemplo, si se necesita una trayectoria recta y otra curva, se pueden llevar a cabo por separado, empezando en ambas con error cero.

2.2. Geometría

El robot se apoya sobre cuatro ruedas distribuidas en dos ejes. Cada rueda es accionada por un motor de corriente continua. Para permitir el giro del robot se utiliza la geometría de dirección de Ackermann. En la figura 2.2 se puede ver un esquema de esa geometría para el eje delantero de un vehículo. En el Robucar ambos ejes utilizan esta disposición, lo que permite varios modos de movimiento que se explican en la sección 2.3.

Se han realizado una serie de medidas sobre el robot, que después se utilizarán para elaborar el modelo. Se muestran en la tabla 2.1. Hay que aclarar que la altura hasta la tabla es la medida desde el suelo hasta la tabla que está encima del robot, como se puede ver en la figura 2.3. El radio teórico de las ruedas es su



(a) Geometría **(b)** Giro

Figura 2.2: La imagen (a) muestra un esquema de una aproximación para el diseño de una geometría de dirección de Ackermann. La imagen (b) muestra el giro de esa geometría.

Medida	Resultado (m)
Longitud del robot	2,90
Anchura del robot	1,30
Altura hasta la tabla	0,93
Altura hasta el punto más alto	2,12
Separación entre ejes	1,20
Longitud de los ejes	1,20
Anchura de las ruedas	0,10
Radio teórico de las ruedas	0,30

Tabla 2.1: Resultados de las mediciones de la geometría del robot.

tamaño aproximado. Es utilizado por el simulador para los cálculos de velocidades necesarias para las ruedas, imitando el comportamiento del robot real, aunque en realidad esos radios varíen según la rueda.

2.2.1. Variación del tamaño de las ruedas

Un problema a la hora de crear el modelo es el tamaño de las ruedas. Para calcular su movimiento el robot considera que su radio es igual para las cuatro y constante. Sin embargo, las cámaras de aire de las ruedas no están siempre perfectamente hinchadas, por lo que los neumáticos están algo aplastados en su parte inferior. Además, las condiciones de presión y de temperatura afectarán a ese tamaño. También el peso que se apoya sobre cada rueda es distinto, y cambia

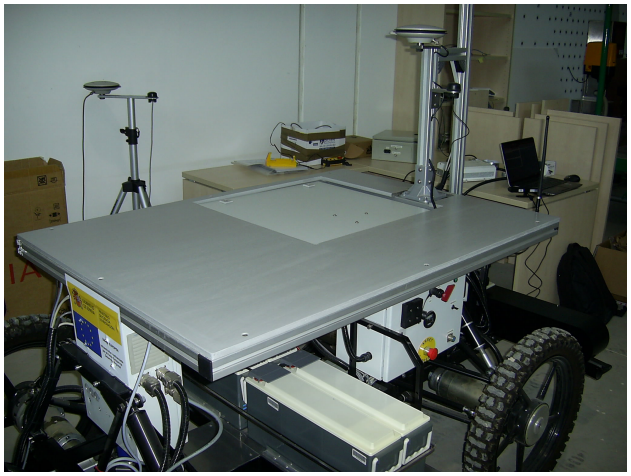


Figura 2.3: Fotografía que muestra la tabla situada encima del robot.

(a) Rueda deshinchada **(b)** Rueda hinchada

Figura 2.4: Comparación entre una rueda tras un período de tiempo sin hinchar y otra recién hinchada.

en función de cómo está cargado el robot y de la posición del brazo robótico. La figura 2.4 muestra una comparación entre una rueda recién hinchada y otra que no ha sido hinchada en varias semanas. Se observa el aplastamiento en la parte inferior, menos pronunciado en la recién hinchada

Ésta es la mayor fuente de error para el modelo. Cambios muy pequeños en los radios de las ruedas de un robot pueden ocasionar grandes errores. Para intentar reducir el efecto, se medirá el tamaño real de cada rueda cuando se realicen las pruebas. Sin embargo, en cuanto cambien las condiciones esas medidas ya no serán correctas. Por lo tanto, es inevitable que se introduzcan errores, y lo único que se puede hacer es intentar acotarlos.

2.3. Cinemática

En este apartado se describe la cinemática del robot en sus tres modos de movimiento: coche, dual y cangrejo o *crab*. En particular, lo que se quiere calcular son las velocidades de giro de las ruedas y el ángulo que deben estar giradas para cumplir con las órdenes de movimiento que se le envién. Estas órdenes tienen dos componentes. La primera es la velocidad lineal del robot y la segunda es el ángulo que tendría una hipotética rueda situada en medio de cada eje. Se explica qué significan exactamente esas órdenes para cada modo. La velocidad lineal del robot está limitada a 4 m/s y el ángulo máximo de giro de la rueda central es de 0,4 radianes.

Para realizar los cálculos de cinemática tanto el robot como sus ruedas son considerados como sólidos rígidos. En el caso de las ruedas la aproximación sería mejor utilizando un sólido deformable. Sin embargo, el objetivo es que el simulador imite al robot real, y éste las trata como sólidos rígidos.

Cuando las ruedas están giradas a la izquierda se considerarán los ángulos como positivos, y viceversa.

2.3.1. Modo coche

En el modo coche el robot mantiene rectas las ruedas traseras y gira las delanteras. Como se puede ver en la figura 2.5, el ángulo que tienen que girar esas ruedas no es el mismo, algo que permite la geometría de Ackermann.

Para este caso, la velocidad lineal del robot es la que corresponde al punto central del eje trasero, v. El ángulo es el que tendría una rueda situada en el centro del eje frontal, θ. A partir de estos dos datos y de la geometría del robot es necesario calcular las velocidades de avance y los ángulos de las cuatro ruedas.

En primer lugar se calcula el radio de giro del robot, R, para el punto central del eje trasero. Para ello se utiliza el triángulo rectángulo formado entre ese punto, el centro de rotación y el punto central del eje trasero, del que se extrae:

$$\tan(\theta) = \frac{S}{R}\,, \tag{2.1}$$

donde la separación entre ejes es S.

Hay que tener en cuenta el caso particular de que el radio de giro sea infinito, es decir, que el robot no gire. Es el caso más sencillo, puesto que implica que el ángulo de todas las ruedas es cero y que la velocidad lineal de todos los puntos es la misma, por lo que se encuentra la solución al problema. La velocidad angular sería cero.

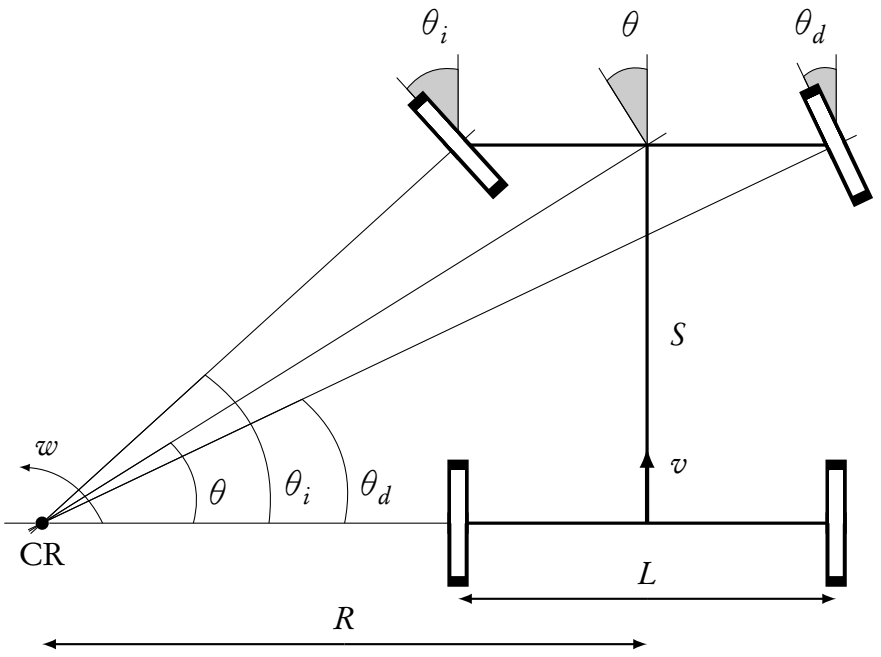


Figura 2.5: Esquema del robot en modo coche, mostrando los ángulos de las ruedas frontales θ_i y θ_d, el de la rueda virtual central θ y el centro de rotación CR.

Para el resto de los casos, se puede calcular la velocidad angular del robot, $w = \dot{\theta}$, a partir de la velocidad lineal del punto central trasero y del radio de giro en ese mismo punto, utilizando:

$$w = \frac{v}{R}. \tag{2.2}$$

Ahora se puede pasar a calcular los ángulos de las ruedas. El de las traseras es cero, por estar en modo coche. Para las ruedas frontales se utilizan los triángulos rectángulos formados por el centro de la rueda, el centro de la rueda trasera del mismo lado y el centro de rotación. Se conocen las longitudes de ambos catetos, siendo uno de ellos la separación entre ejes S y el otro es el radio de giro del punto central del eje trasero, sumando o restando la mitad de la longitud del eje L.

Las siguientes ecuaciones indican el ángulo de giro de la rueda delantera izquierda, θ_i, y derecha θ_d.

$$\theta_i = \tan^{-1}\left(\frac{S}{R - \frac{L}{2}}\right), \tag{2.3}$$

$$\theta_d = \tan^{-1}\left(\frac{S}{R+\frac{L}{2}}\right). \tag{2.4}$$

En este contexto un radio de giro negativo indica un giro a la derecha. Hay que recordar que el radio de giro R sigue siendo el del centro del eje trasero del robot.

Conocidos los ángulos de las ruedas, resta saber sus velocidades. Dado que sabemos la velocidad lineal de un punto y la velocidad angular del robot, se puede calcular la velocidad de cualquier punto, al ser un sólido rígido. La siguiente ecuación expresa esa relación:

$$\mathbf{v} = \mathbf{v}_0 + \omega \times \mathbf{r}\,, \tag{2.5}$$

donde $\mathbf{v}$ es la velocidad de un punto, $\mathbf{v}_0$ es la velocidad de otro punto, $\mathbf{r}$ es el vector que une ambos puntos y ω es la velocidad angular del sólido.

Para saber la velocidad lineal a la que debe avanzar cada rueda se calculará el módulo de los vectores velocidad obtenidos, puesto que ya están orientadas en la dirección del movimiento:

$$v_{f,i} = \sqrt{\left(v - w \cdot \frac{L}{2}\right)^2 + (w \cdot S)^2}\,, \tag{2.6}$$

$$v_{f,d} = \sqrt{\left(v + w \cdot \frac{L}{2}\right)^2 + (w \cdot S)^2}\,, \tag{2.7}$$

$$v_{t,i} = v - w \cdot \frac{L}{2}\,, \tag{2.8}$$

$$v_{t,d} = v + w \cdot \frac{L}{2}\,, \tag{2.9}$$

donde $v_{f,i}$ es la velocidad lineal de la rueda frontal izquierda, $v_{f,d}$ la de la frontal derecha, $v_{t,i}$ la de la trasera izquierda y $v_{t,d}$ la de la trasera derecha. Hay que destacar que estas ecuaciones dan velocidades positivas, el signo debe ser invertido para movimientos hacia atrás.

Por último, es necesario saber a qué velocidad angular deben girar cada rueda, conocido su radio r. La siguiente ecuación muestra esa velocidad para una rueda cualquiera, w_r. La velocidad lineal requerida a esa rueda es v_r:

$$w_r = \frac{v_r}{r}\,. \tag{2.10}$$

Particularizando para cada rueda se obtiene:

$$w_{f,i} = \frac{v_{f,i}}{r_{f,i}}\,, \tag{2.11}$$

$$w_{f,d} = \frac{v_{f,d}}{r_{f,d}}\,, \tag{2.12}$$

$$w_{t,i} = \frac{v_{t,i}}{r_{t,i}}\,, \tag{2.13}$$

$$w_{t,d} = \frac{v_{t,d}}{r_{t,d}}\,. \tag{2.14}$$

2.3.2. Modo dual

Para lograr radios de giro menores, también se puede modificar la posición de las ruedas traseras, que también están configuradas según la geometría de Ackermann. La figura 2.6 muestra un esquema del robot en este modo. Se puede observar que los ángulos de las ruedas traseras son los opuestos de sus correspodientes ruedas delanteras.

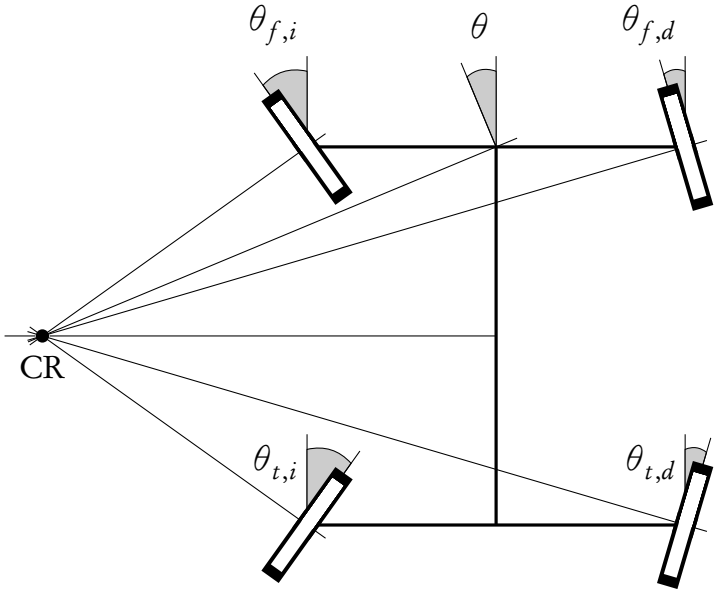


Figura 2.6: Esquema del robot en modo dual, mostrando los ángulos de las ruedas frontales $\theta_{f,i}$ y $\theta_{f,d}$, los de las ruedas traseras $\theta_{t,i}$ y $\theta_{t,d}$, el de la rueda virtual central θ y el centro de rotación CR.

En este caso, la velocidad lineal del robot, v, que se recibe como orden es la del punto central. Se entiende que ese punto es el que equidista de los centros de ambos ejes, no el centro de gravedad del robot. El ángulo requerido, θ, sigue siendo el de la hipotética rueda situada en el centro del eje frontal, que también correspondería al opuesto de la rueda que estaría en el centro del eje trasero.

La siguiente ecuación muestra cómo se puede obtener el radio de giro para el punto central del robot, con el triángulo rectángulo formado entre ese punto, el punto central del eje delantero y el centro de rotación CR:

$$\tan(\theta) = \frac{S}{2R} \,. \tag{2.15}$$

Comparando con la ecuación 2.1 del modo coche, se puede ver que para el mismo ángulo el radio de giro que se obtiene es la mitad.

De nuevo, si el radio de giro es infinito el robot va recto, su velocidad angular será nula y los ángulos de las vuatro ruedas serán cero.

En el resto de las situaciones, la velocidad angular se calcula con la ecuación 2.2, igual que en el modo coche.

Para el cálculo de los ángulos de las ruedas delanteras se utilizan los triángulos formados por el centro del eje frontal, el centro de rotación y el punto situado debajo de la rueda a la altura del centro del robot. Los de las ruedas traseras son simplemente los opuestos de su correspondiente rueda delantera. Se muestra cómo se hallan el ángulo de la rueda delantera izquierda, $\theta_{f,i}$, y el de la derecha, $\theta_{f,d}$:

$$\theta_{f,i} = \tan^{-1}\left(\frac{S}{2\left(R-\frac{L}{2}\right)}\right), \tag{2.16}$$

$$\theta_{f,d} = \tan^{-1}\left(\frac{S}{2\left(R+\frac{L}{2}\right)}\right). \tag{2.17}$$

Conocidos los ángulos, se obtienen las velocidades de las ruedas utilizando la ecuación 2.5 y calculando sus módulos, al igual que en el modo coche. Por simetría, las velocidades de las ruedas traseras son iguales que las de su correpondiente rueda delantera. Se obtiene la velocidad de la rueda delantera izquierda, $v_{f,i}$, y la de la derecha, $v_{f,d}$:

$$v_{f,i} = \sqrt{\left(v - w\cdot\frac{L}{2}\right)^2 + \left(w\cdot\frac{S}{2}\right)^2}\,, \tag{2.18}$$

$$v_{f,d} = \sqrt{\left(v + w \cdot \frac{L}{2}\right)^2 + \left(w \cdot \frac{S}{2}\right)^2}. \tag{2.19}$$

De nuevo, la velocidad angular de las ruedas se calcula mediante la ecuación 2.10.

2.3.3. Modo *crab*

El modo *crab*, o cangrejo, permite que el robot se mueva en diagonal, recordando en cierto modo al andar lateral de estos animales. En teoría, las cuatro ruedas deberían girar el mismo ángulo. Sin embargo, la misma geometría de Ackermann que permitía colocar todas las ruedas en el ángulo correcto en los dos modos anteriores impide lograr ese objetivo. Por consiguiente, únicamente se podrá lograr una aproximación y las ruedas se verán forzadas a deslizar. Esta es la razón por la que este modo de movimiento debería ser considerado como secundario y se recomienda evitar su uso en la medida de lo posible.

La figura 2.7 contiene un esquema de este modo. Los ángulos de las ruedas se pueden obtener utilizando centros de rotación virtuales para cada eje. En realidad no existe rotación, pero sirven para garantizar que la posición de las ruedas cumple con la geometría de Ackermann, como están construidas.

La velocidad lineal requerida en este modo se considera como la velocidad a la que debe avanzar el robot. En este modo no gira, por lo que la velocidad es la misma para todos sus puntos. El ángulo que se requiere es el de las ruedas virtuales centrales, θ.

Las ruedas de cada lado estarán giradas el mismo ángulo, por lo que bastará con calcular los del eje delantero. Se puede tomar un radio de giro virtual R igual que el del modo coche de la ecuación 2.1, ya que son conocidas la separación entre ejes S y el ángulo virtual requerido θ. Acto seguido se aplica la ecuación 2.3 para calcular los ángulos de las ruedas izquierdas y la ecuación 2.4 para las derechas.

Por supuesto, para pasar de la velocidad de avance de una rueda a su velocidad angular se vuelve a utilizar la ecuación 2.10.

2.4. Dinámica

Un problema que se encuentra a la hora de caracterizar el robot es la ausencia de información sobre algunos de sus parámetros físicos. Por supuesto, su geometría se puede medir, pero el robot no ha sido pesado ni esa información ha sido proporcionada por el fabricante. Únicamente se sabe que su masa está en torno a los 250 kg. Tampoco se tiene mucha información sobre su tensor de inercia, es

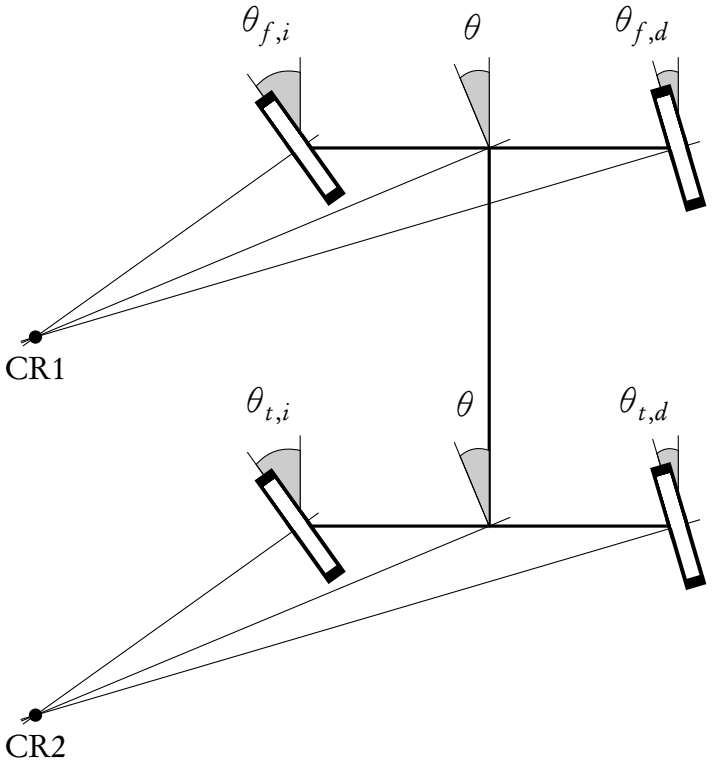


Figura 2.7: Esquema del robot en modo *crab*, mostrando los ángulos de las ruedas frontales $\theta_{f,i}$ y $\theta_{f,d}$, los de las ruedas traseras $\theta_{t,i}$ y $\theta_{t,d}$, el de la rueda virtual central θ y los centros de rotación virtuales CR1 y CR2.

decir, sobre cómo está distribuida esa masa. El brazo robótico, con una masa en torno a los 50 kg, está situado en la parte delantera izquierda del robot y hace que esa sea su parte más pesada, pero esa es toda la información disponible.

Por supuesto, la fuerza más importante a la que está sometido el robot es el peso. Al transmitir ese peso al suelo se produce una reacción igual y de sentido opuesto, según la tercera ley de Newton. Esta reacción, normal al plano del suelo, se reparte entre las cuatro ruedas. Si la distribución de masa del robot fuera uniforme sería igual para todas, pero al no serlo dependerá de esa distribución.

De esas fuerzas normales depende la fuerza de fricción sobre el suelo. Según el modelo de Coulomb, utilizado por el simulador Gazebo, esa fuerza es menor o igual que un coeficiente de rozamiento multiplicado por la normal, según la ecuación:

$$F_r \leq \mu F_n \tag{2.20}$$

donde F_r es la fuerza de rozamiento, μ es el coeficiente de rozamiento y F_n es la

fuerza normal al suelo. Esta fuerza de rozamiento se produce en la dirección del movimiento y se opone a éste.

Se forma un cono de fricción con la suma de la fuerza normal y el máximo rozamiento. Por debajo de ese cono, la fuerza de rozamiento es insuficiente para evitar el movimiento. El simulador utiliza una aproximación de este modelo, como se explica en la sección 3.1.2.

Para una rueda, la situación ideal es que este rozamiento sea lo más elevado posible, evitando el deslizamiento. Para una rueda que contactara con el suelo en un punto, el óptimo sería rozamiento infinito. Las del robot tienen una geometría de contacto más compleja, debido al aplastamiento de las ruedas mostrado en la figura 2.4. Además, hay que tener en cuenta que no es constante, depende del hinchado de la rueda en ese momento. Incluso cambiará cuando el robot acelere, al aparecer otras fuerzas además del peso.

El coeficiente de rozamiento entre el neumático de goma y el suelo también puede variar, según el tipo de suelo, la velocidad, la temperatura, la presión y la geometría de contacto, entre otras variables ([9], [10]). Un valor razonable para el coeficiente μ entre goma y asfalto es 0,9, según [10].

El par motor que se puede proporcionar a las ruedas no supone ninguna limitación para la velocidad que puede alcanzar el robot, ya que ésta no puede superar los 4 m/s por el *software*, por debajo de la que podría llegar.

CAPÍTULO 3

IMPLEMENTACIÓN EN EL SIMULADOR

En este capítulo se explica cómo se implementa el modelo del robot en el simulador Gazebo. En primer lugar, se crea un robot con unas características similares al Robucar, sin buscar replicar su comportamiento exacto. De esta forma, se puede comprobar que los controles creados para el robot simulado funcionan correctamente. Una vez se consigue, se pasa pasar a intentar acercar el comportamiento del robot simulado al del real lo máximo posible.

Además, debe funcionar utilizando exactamente las mismas órdenes que el robot real. De esta forma, se podrán utilizar los mismos programas sin necesidad de ninguna modificación.

3.1. *Software* utilizado

Esta sección trata sobre el *software* más importante utilizado para poder simular el robot. En particular, se explica brevemente cómo funciona el simulador Gazebo y cómo interactúa con el entorno ROS. Se presta especial atención a los elementos más importantes para este proyecto.

3.1.1. ROS

ROS son las siglas de *Robot Operating System*, Sistema Operativo Robótico [2]. Como su nombre indica, es un sistema operativo que permite controlar uno o varios robots, además de proporcionar una serie de paquetes con los que se pueden realizar tareas comunes como, por ejemplo, la planificación de rutas. Estos paquetes son contribuciones de usuarios y, como se ha hecho en este proyecto, se pueden modificar o añadir nuevos.

En realidad, no es un sistema operativo en el sentido tradicional del término, sino que proporciona una capa de comunicación por encima de los sistemas operativos de los equipos sobre los que funcione. En este proyecto se utiliza un único ordenador con Ubuntu 12.04.

El núcleo de ROS es código abierto, al igual que la mayoría de sus paquetes. Esto es muy importante, ya que permite ver cómo se han realizado simuladores de otros robots como el PR2 [3] y el TurtleBot [4].

Hay cuatro elementos que son imprescindibles para entender cómo funciona ROS: los nodos, los mensajes, los *topics* y los servicios.

Los nodos son los procesos que se encargan de realizar los cálculos. Normalmente, un sistema de control de un robot se compondrá de varios nodos. Por ejemplo, un nodo podría encargarse de la localización del robot, otro de recibir información de un sensor láser y otro de mover las ruedas. El sistema se podría ampliar para incluir muchas otras funciones.

Existe un nodo central, llamado Maestro o *Master*, que se encarga de que los nodos puedan encontrarse unos a otros, intercambiar mensajes e invocar servicios.

Los nodos se comunican entre sí enviándose unos a otros mensajes. Estos consisten en varios campos con un tipo de dato, como pueda ser un número entero, un valor lógico o una cadena de caracteres. Los mensajes pueden incluir estructuras anidadas. Por ejemplo, un mensaje de situación de un robot podría consistir en su posición y su orientación. A su vez, la posición consistiría en tres números reales representando las coordenadas cartesianas y la orientación consistiría en otros tres números reales representando los ángulos de Euler.

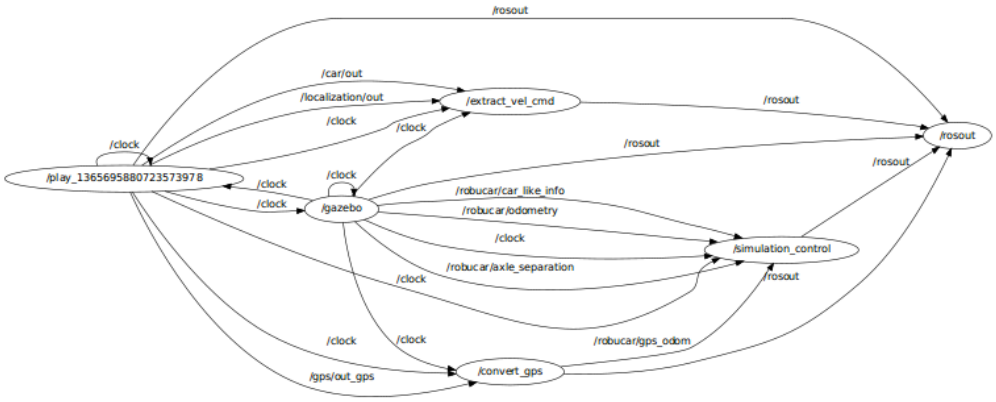
Se pueden utilizar varios lenguajes de programación para programar los nodos. Para este proyecto se ha utilizado C++, pero se podrían utilizar otros lenguajes como Python.

Para la comunicación asíncrona entre nodos se utilizan *topics*, o temas. Los *topics* tienen un nombre que los identifica. Cuando un nodo quiere enviar un mensaje, lo publica en un *topic*. Los nodos interesados en un tipo de mensaje se suscribirán al *topic* correspondiente. Varios nodos pueden publicar y suscribirse al mismo *topic*, y un nodo puede publicar y suscribirse a distintos *topics*.

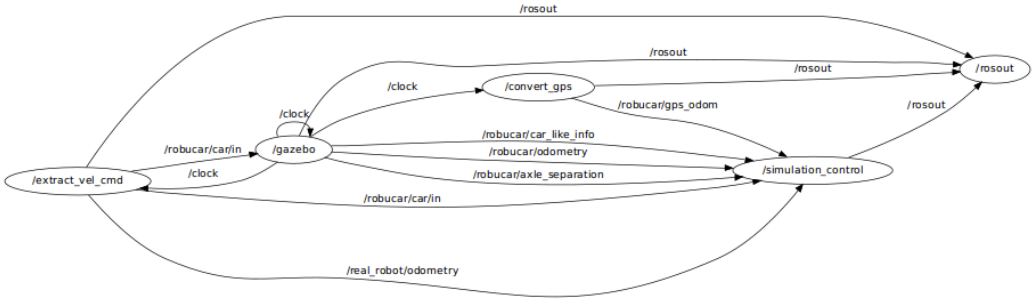
Cuando es necesaria una comunicación síncrona, se utilizan servicios. Un nodo envía una petición a otro en forma de mensaje y éste contesta con otro mensaje. Los nodos anuncian los servicios que ofrecen, y al solicitarlos el otro nodo espera a recibir una respuesta.

Las *bags* o *rosbags*, bolsas, son un formato para guardar y reproducir mensajes. En este proyecto se utilizarán para guardar información del robot real y poder compararlo con el simulado.

El robot Robucar puede ser controlado mediante ROS, y el simulador Gazebo puede funcionar como un nodo. Esto permite que tanto el robot real como el



(a) Reproducción de la información de una prueba.



(b) Procesado de la información de una prueba.

Figura 3.1: Funcionamiento de ROS con varios nodos. Las flechas indican envío de mensajes mediante *topics*.

simulado funcionen con el mismo tipo de mensaje publicado en un tema con el mismo nombre, facilitando al máximo el uso del simulador.

La figura 3.1 muestra un ejemplo de ROS funcionando con varios nodos, durante una de las pruebas que se describen en el capítulo 4. Las flechas indican el envío de mensajes a través de un determinado *topic*, desde el nodo que publica hasta el que recibe. Se puede observar que varias flechas tienen el mismo nombre, lo que indica que más de un nodo publica o se suscribe a ese *topic*.

La subfigura 3.1a representa la primera fase de una prueba, en la que se reproduce la información de una prueba real almacenada en una *rosbag* y se guarda en un nodo. En la segunda fase, mostrada en la subfigura 3.1b, esa información se vuelve a reproducir como órdenes para el robot real. Otro nodo recoge todos los datos que necesita para proporcionar las gráficas y tablas del capítulo 4 y el anexo E.

Este proceso se explica con más detalle en el anexo D.

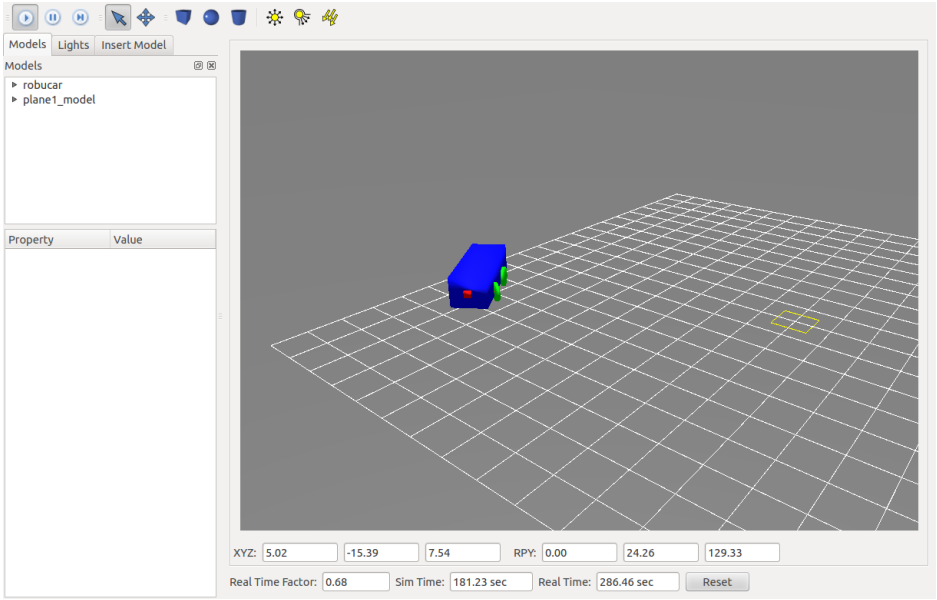


Figura 3.2: Ejemplo de funcionamiento de la interfaz gráfica de Gazebo durante una prueba del simulador.

La versión de ROS utilizada para este proyecto es la quinta, Fuerte [11], aunque se empezó utilizando una anterior, Diamondback [12], y fue necesario actualizar alguna parte de los programas escritos hasta entonces.

Para una explicación más extensa sobre cómo funciona ROS, se puede consultar [13].

3.1.2. Gazebo

Gazebo es un programa que permite simular varios robots, sensores y objetos en un entorno tridimensional. Puede generar información realista para los sensores e incluye un simulador de dinámica de sólidos rígidos que puede recrear interacciones plausibles entre ellos [1].

Puede funcionar como un nodo de ROS, lo que permite el control de la simulación mediante mensajes, además de recibir información a través de ellos. Adicionalmente, y teniendo en cuenta que el robot real también se controla utilizando ROS, esos mensajes pueden ser exactamente los mismos para el robot simulado, simplificando su uso.

Gazebo también incluye una interfaz gráfica que permite visualizar un escenario y los movimientos de los robots. La figura 3.2 la muestra durante una de las pruebas del simulador.

Tanto los modelos de los robots como los escenarios se describen utilizando

URDF o SDF como formato. URDF son las siglas de *Unified Robot Description Format*, formato de descripción de robots unificado y SDF las de *Simulation Description Format*, formato de descripción de simulación. Ambos son ficheros XML. Para este proyecto se ha usado URDF, al ser el que único que estaba disponible cuando se inició.

En ambos formatos, un robot se compone de *joints* y *links*, articulaciones y enlaces. Las articulaciones son los sólidos rígidos de los que se compone el robot y los enlaces sus uniones.

Para definir un sólido en URDF son necesarios tres elementos. El primero contiene información sobre su inercia. En particular, su masa y cómo está distribuida según un tensor de inercia. Los otros dos elementos contienen la geometría del objeto, pero se distingue entre la que se ve y la que se utiliza para calcular colisiones entre sólidos. De esta forma, se puede utilizar una geometría de visión más compleja para hacer que el robot parezca más realista mientras que la de colisión se mantiene más sencilla para facilitar los cálculos y no ralentizar la simulación. Esta característica se aprovecha en el capítulo 5 para cambiar el aspecto del robot sin modificar su comportamiento.

Las uniones entre sólidos pueden ser de varios tipos, según cómo se limiten sus grados de libertad. Por ejemplo, pueden permitir el giro en una dirección o combinar dos sólidos en uno eliminando todos los grados de libertad. También se puede limitar el rango de movimiento, como podría ser permitir una rotación entre 0°y 90°.

Para definir el de rozamiento de las ruedas de un robot con el suelo es necesario modificar el escenario. Este escenario puede incluir varios planos distintos, permitiendo simular varios terrenos con distintos coeficientes de rozamiento.

Gazebo utiliza el modelo de la pirámide de fricción para simular el rozamiento, una simplificación del modelo del cono de fricción para facilitar los cálculos [14]. Es necesario proporcionarle los coeficientes de fricción seca en las direcciones principal y secundaria de fricción según la pirámide y esa dirección principal.

Hay que destacar que actualizar a ROS Fuerte implicó cambiar a la versión 1.0 de Gazebo, con un cambio en la interfaz de programación, por lo que fue necesario modificar buena parte del código escrito y volver a ajustar algunos parámetros.

En [15] se explica con más detalle qué es Gazebo y sus características más importantes.

3.2. Modelado del robot

Como se ha expuesto en la sección 3.1.2, la masa del robot, el modelo de colisión y el aspecto visual del robot son elementos distintos en una descripción

URDF que se introducen por separado. En esta sección se explica cómo se han modelado los dos primeros, mientras que el capítulo 5 trata sobre el último.

En la sección 2.1 se ha hablado de los problemas existentes para conocer algunos datos sobre el robot, lo que implica que el modelo no se ajustará perfectamente a la realidad. Con el objetivo de crear un simulador más realista, en la sección 3.4 se estudiará qué parámetros son más importantes y cuáles tienen un efecto menor sobre la cinemática y dinámica del robot simulado. Después, los más importantes serán ajustados utilizando información de pruebas reales en el capítulo 4.

3.2.1. Distribución de masa

La masa está en torno a los 250 kg, como se explica en la sección 2.1, y ése es el valor que se elije para el modelo.

Se asumirá que la mayor parte de la masa, 200 kg, está distribuida uniformemente en un ortoedro. Sus dimensiones serán la altura hasta la tabla, la longitud, y la anchura de la tabla 2.1. Los 50 kg restantes se utilizarán para simular la presencia del brazo mecánico y se colocarán en un punto en la esquina delantera izquierda, que será incluido en el modelo como un objeto distinto. De esta manerá será muy sencillo reajustar su posición.

El tensor de inercia, necesario para la descripción del robot en URDF, se representa como una matriz simétrica 3x3:

$$\mathbf{I} = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{pmatrix}. \tag{3.1}$$

Existe un sistema de referencia cartesiano para el cual los elementos fuera de la diagonal principal serán nulos, el formado por las tres direcciones de simetría del cuboide:

$$\mathbf{I} = \begin{pmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{pmatrix}. \tag{3.2}$$

Los elementos de la diagonal principal son los momentos principales de inercia. Para calcular el momento de inercia de un sólido continuo en torno a un eje se utiliza la expresión:

$$I_x = \int_V \mathbf{r}^2 \, dm = \int_V \rho(\mathbf{r}) \mathbf{r}^2 \, dV, \tag{3.3}$$

donde m es la masa del sólido, $\mathbf{r}$ es el radio vector de ese punto, es decir, el vector que lo une con el eje de rotación, y $\rho(\mathbf{r})$ es la densidad de ese punto. La integral se evalúa en todo el volumen V del sólido.

Para un ortoedro de densidad constante con longitud l, anchura a y altura h que rota en torno al eje x la expresión se convierte en:

$$I_x = \rho \int_{x=-\frac{l}{2}}^{x=\frac{l}{2}} \int_{y=-\frac{a}{2}}^{y=\frac{a}{2}} \int_{z=-\frac{h}{2}}^{z=\frac{h}{2}} \left(y^2+z^2\right)\, dx\, dy\, dz. \tag{3.4}$$

En primer lugar se integra sobre z:

$$I_x = \rho \int_{x=-\frac{l}{2}}^{x=\frac{l}{2}} dx \int_{y=-\frac{a}{2}}^{y=\frac{a}{2}} \left(y^2 h + \frac{h^3}{12}\right) dy. \tag{3.5}$$

Después sobre y:

$$I_x = \rho \int_{x=-\frac{l}{2}}^{x=\frac{l}{2}} \left(\frac{a^3}{12}h + \frac{h^3}{12}a\right) dx. \tag{3.6}$$

Y por último sobre x:

$$I_x = \rho \left(\frac{a^3}{12}h + \frac{h^3}{12}a\right) l = \frac{1}{12} m \left(a^2+h^2\right). \tag{3.7}$$

Se sigue un razonamiento análogo para los otros dos momentos principales de inercia:

$$I_y = \frac{1}{12} m \left(l^2+h^2\right), \tag{3.8}$$

$$I_z = \frac{1}{12} m \left(l^2+a^2\right). \tag{3.9}$$

Lo que nos permite calcular, utilizando los datos de la tabla 2.1, los momentos de inercia del robot.

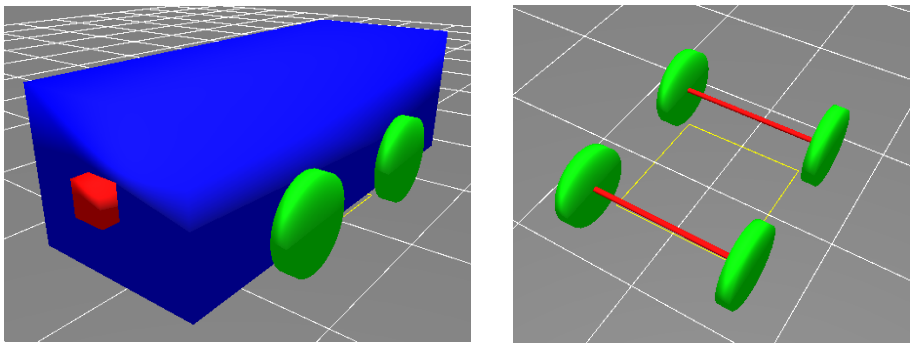
Para los demás sólidos del modelo se utilizan valores muy pequeños, ya que Gazebo no permite que sean cero.

3.2.2. Estructura

En este apartado se describen los sólidos creados para el modelo de colisión del robot. Como se explica en la sección 3.1.2, el modelo de colisión contiene la geometría del robot y se utiliza para las interacciones con su entorno, pero no

Momentos	Valor (kg m^2)
I_x	42,58
I_y	154,58
I_z	168,33

Tabla 3.1: Momentos principales de inercia del modelo del robot.



(a) Robot **(b)** Ruedas

Figura 3.3: En la imagen (a) se puede ver el modelo completo del robot y en la (b) las ruedas con sus ejes. El aspecto visual muestra el modelo de colisión.

tiene por qué coincidir con su visualización. En este capítulo se hace coincidir el modelo visual con el de colisión. Posteriormente, en el capítulo 5, se intenta imitar el aspecto real.

Se simplifica la mayor parte de la estructura del robot como un ortoedro, como se puede ver en la figura 3.3a. Un pequeño cubo rojo es incluido en la parte delantera para identificarla. Este modelo simplificado permite acelerar la simulación.

Se añaden dos ejes en la parte inferior, que unen el cuerpo del robot con las ruedas. Una unión en Gazebo no permite el giro en dos direcciones distintas, así que para cada rueda hay un pequeña pieza que gira sobre el eje y se encarga de orientarlas. Se les ha llamado rodamientos, al recordar en cierto modo a la función de éstos. Se muestran en la figura 3.4.

El giro de las propias ruedas, montadas sobre los rodamientos, es el que permite el movimiento de avance del robot.

El control por *software* se encarga de que las posiciones de las ruedas mantengan la geometría de Ackermann, en vez de recrearla en el modelo. De esta manera se simplifica y se facilita la creación de modelos de otros robots distintos en el futuro. Además, mientras el control se asegure de que los ángulos sean correctos

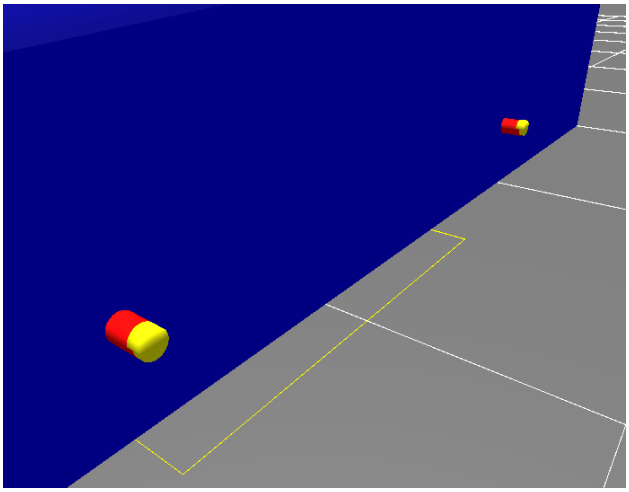


Figura 3.4: Piezas, llamadas rodamientos, sobre las que giran las ruedas permitiendo su orientación, en amarillo. Están situados en los dos extremos de cada eje.

no comporta ninguna desventaja.

También se limitan así los ángulos que pueden girar las ruedas. El motor de física del simulador da problemas al acercarse a los límites definidos en el modelo. Por ejemplo, el robot puede empezar a saltar, algo completamente absurdo. Las restricciones de esos ángulos se fijan suficientemente lejos como para evitar que se acerque en ningún momento.

3.3. Control del robot

Básicamente, hay dos opciones disponibles para controlar un modelo en Gazebo:

- La primera es crear un nodo de ROS que publique mensajes con todas las órdenes para las ruedas del robot, tanto para el control de su posición como el de la velocidad de giro. A su vez, ese nodo leería la información de los sólidos del robot publicada por Gazebo y extraería la odometría.

- La otra opción es crear un *plugin* para Gazebo, que es un fragmento de código que se inserta dentro del programa. Permite controlar prácticamente cualquier aspecto de la simulación. Además, garantiza que se ejecutará una vez en cada paso de simulación, utilizando el estado del robot en ese instante. Es un poco más complejo que la opción anterior. Sin embargo, para un caso

como el actual, que necesita controlar cuatro ruedas distintas, la diferencia es mínima.

El *plugin* también es más rápido que la primera opción. El envío de mensajes requiere tiempo, así que con el nodo independiente no se garantiza que en cada momento se estén dando las órdenes adecuadas al estado del robot, o que éstas lleguen suficientemente pronto. No supondría un inconveniente para la velocidad de giro de las ruedas, que cambia muy poco comparado con el tiempo de cada paso del simulador; pero sí para el control de su posición, que utilizará un controlador PID. La única ventaja que presenta frente al *plugin* es que es más sencilla su implementación, ya que consiste simplemente en un nodo de ROS.

Se ha elegido utilizar el *plugin*, al considerar que las ventajas que supone su uso superan con creces los problemas. Los mensajes que se usan para la comunicación entre Gazebo y el resto de los nodos de ROS se describen en el anexo B.

El control se actualiza cada décima de segundo en la simulación, un tiempo suficientemente pequeño para proporcionar una buena precisión sin ralentizar demasiado el simulador. En caso necesario, basta con cambiar un valor para reducirlo y aumentar la precisión de la simulación o incrementarlo para hacerla más rápida.

Al recibir las órdenes, se calcula la velocidad de giro y la posición de cada una de las ruedas a partir de las ecuaciones cinemáticas expuestas en la sección 2.3.

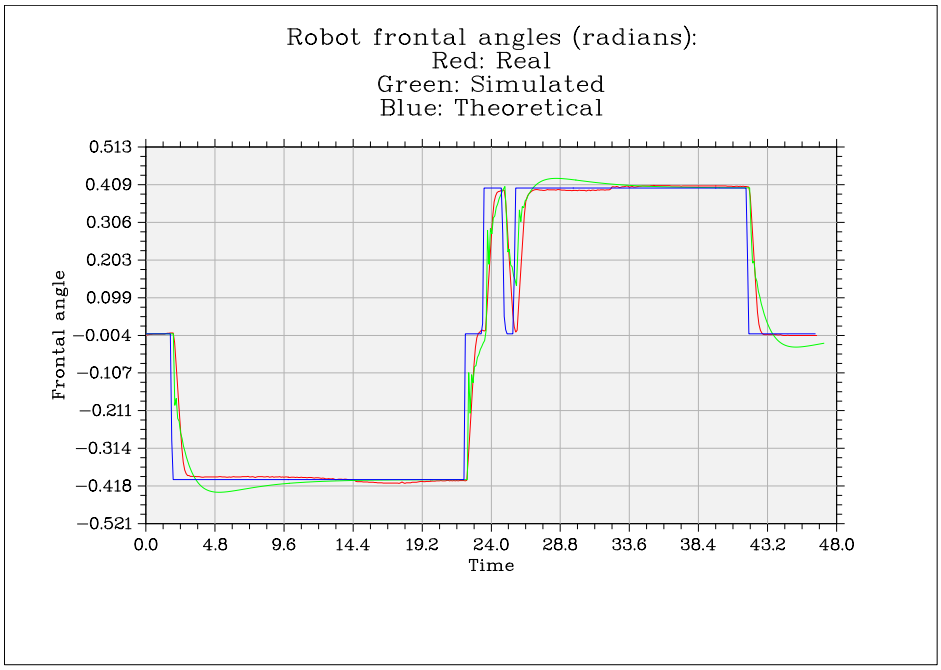
Para controlar el movimiento de los sólidos en Gazebo hay dos opciones: aplicar una fuerza o fijar una velocidad. Teniendo en cuenta las incertidumbres existentes en la distribución de masa, se utilizarán velocidades, paliando su efecto. También se limita la fuerza máxima que se puede aplicar para alcanzar esa velocidad, evitando discontinuidades en ella al restringir la aceleración.

De esta forma, el control de la velocidad de giro de las ruedas es directo, simplemente se fija su velocidad.

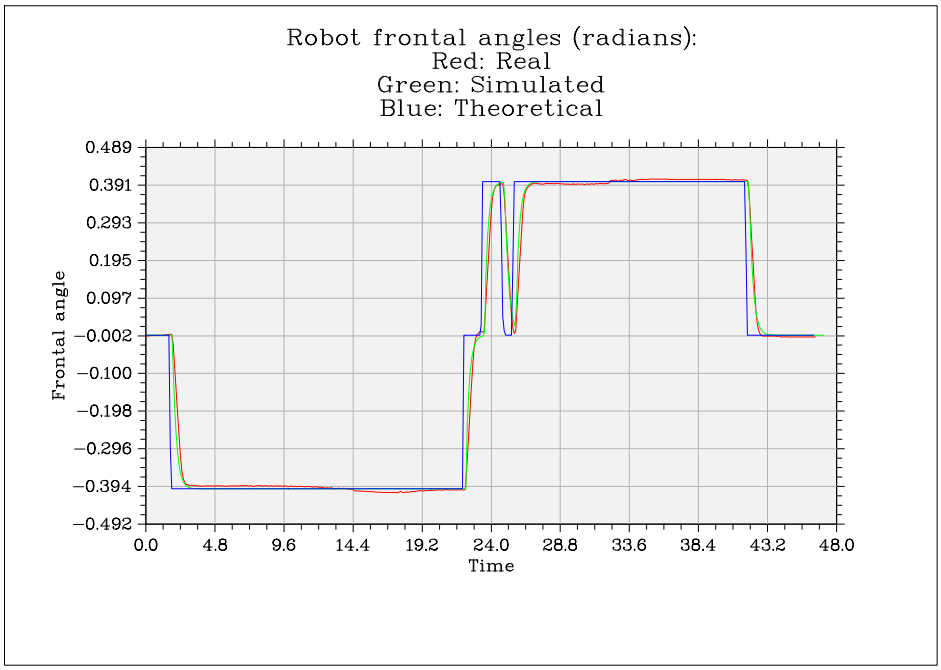
La posición es más compleja, ya que no se puede actuar sobre ella directamente. Para ello se añade un controlador proporcional integral derivativo (PID), que ajusta la velocidad para cumplir con la consigna que se le requiera. La acción integral es necesaria para que el error de posición sea cero.

Para el ajuste de las constantes del controlador se prueban distintos valores hasta obtener un comportamiento similar al del robot real. La figura 3.5 muestra el comportamiento de la rueda virtual central antes y después del ajuste del PID, utilizando las órdenes y el comportamiento de una prueba real. Se puede observar que la sobreoscilación desaparece y se reduce el tiempo de respuesta, ajustándose mejor a la consigna.

Se han incluido las constantes del PID en la descripción URDF del robot para facilitar su posterior modificación, en caso de que sea necesario, según se explica



(a) Antes del ajuste



(b) Después del ajuste

Figura 3.5: Se muestra la evolución del ángulo de la rueda central virtual antes (a) y después (b) de ajustar. En azul está el ángulo requerido, en rojo el del robot real y en verde el del simulado.

en el anexo F.

Dado que el *plugin* tiene acceso a la posición y velocidad del robot en todo momento, se puede publicar en ROS toda esa información. Para ello se utilizan los mensajes *Odometry*, *CarLikeInfo* y *AckermannDriveStamped* del anexo B.

3.4. Análisis de sensibilidad

Antes de intentar ajustar el robot simulado al comportamiento del real, se va a comprobar que realmente el factor más influyente en el simulador es el tamaño de las ruedas, mientras que la importancia de otros parámetros como la masa, el tensor de inercia y el rozamiento con el suelo es mucho menor.

Para ello, se realizan una serie de pruebas en el simulador, cambiando esos parámetros y comprobando su efecto. Esas pruebas se describen en el anexo C.

Hay que tener en cuenta que esto no implica que en la realidad esos factores no sean importantes en la realidad, únicamente se comprueba su influencia en el robot simulado. De esta forma, se sabrá qué parámetros se pueden modificar para ajustar la simulación al robot real en el capítulo 4.

Los resultados indican que los únicos factores que tienen un efecto importante en la simulación son los radios de cada rueda. Esto se debe a que el control se realiza fijando velocidades, proporcionando la fuerza necesaria para alcanzarla.

El rozamiento, que es un factor muy importante en la realidad, no puede ser modelado correctamente utilizando el motor de física de Gazebo. La geometría de contacto varía según la rueda y cómo esté hinchada, la distribución de peso sobre las ruedas no es uniforme y cada terreno tendría un coeficiente de rozamiento distinto. Además, se ha comprobado que cambiar únicamente ese coeficiente no es suficiente. Por lo tanto, es necesario tenerlo en cuenta de otra forma, como se explica en la sección 4.2.3.

La tabla 3.2 muestra un resumen de los distintos parámetros del modelo que deben ser ajustados, incluyendo una valoración cualitativa de su importancia.

El marcado con las tres cruces, los radios de las ruedas, es un parámetro muy importante. Pequeños cambios producen grandes variaciones en el comportamiento. Entre los que tienen asignadas dos cruces, están los parámetros de la geometría de los ejes, muy relevantes, pero que pueden ser medidos directamente y que permanecen prácticamente constantes. También se considera que las velocidades máximas y el ángulo máximo de giro tienen una relevancia media, dado que únicamente influyen al acercarse el robot a esos límites. Además, están bien definidos en las especificaciones del robot.

El último con dos cruces son las constantes del PID, que controlan los ángulos de las ruedas. Que no estén bien ajustadas implicaría un peor comportamiento de esos ángulos, pero su efecto únicamente se nota mucho cuando se producen

	Parámetro	Importancia
Geometría	Radios de las ruedas	+++
	Separación entre ejes	++
	Longitud de eje	++
Cinemática	Velocidad lineal máxima	++
	Velocidad angular máxima	++
	Ángulo máximo de giro	++
Dinámica	Masa del robot	+
	Tensor de inercia	+
	Coeficiente de rozamiento	+
	Constantes PID	++
	Fuerza máxima ruedas	+

Tabla 3.2: Resumen de parámetros que se deben ajustan.

cambios bruscos en ellos.

Los demás parámetros tienen una importancia muy reducida en el comportamiento de la simulación.

CAPÍTULO 4

PRUEBAS Y AJUSTE

En el capítulo 3 se implementa en el simulador Gazebo un modelo del robot utilizando sus parámetros reales. Sin embargo, esto no implica que el comportamiento en la simulacion se ajuste a la realidad, tanto por las simplificaciones y aproximaciones del modelo como por las características de Gazebo.

Por lo tanto, en este capítulo se realizan una serie de pruebas para comparar las respuestas tanto del robot real como del simulado a las mismas órdenes, con la idea de ajustarlo lo máximo posible. Posteriormente, se evalúan los resultados de este ajuste.

Hay que mencionar que en estos momentos el control de las ruedas traseras del robot real no funciona, por lo que únicamente se puede utilizar en modo coche. Por consiguiente, ése es el modo del funcionamiento que se ha utilizado para realizar las pruebas y ajustar el simulador.

4.1. Pruebas

Se han realizado una serie de pruebas utilizando el robot real, guardando la información que proporciona en *rosbags*. Después se usa para reproducir las mismas órdenes en el simulador. Entonces se pueden comparar las trayectorias y velocidades, buscar las diferencias e intentar corregirlas. Los resultados de estas pruebas se recogen en el anexo E.

Se llevan a cabo seis pruebas distintas para poder ajustar el robot. Dos de ellas son trayectorias rectilíneas efectuadas a velocidades distintas. Otras dos son simplemente giros con velocidades crecientes, cada uno en un sentido. Por último, se efectúan dos ochos, a velocidades distintas. Todas las pruebas se realizan sobre un suelo de grava. Además, se repiten tres de esas pruebas para comprobar el resultado del ajuste. Por último, se realizan dos pruebas más para observar

cómo funciona el simulador en una condiciones para las que no ha sido ajustado específicamente.

Esta serie de pruebas permiten encontrar errores tanto en la trayectoria del robot como en sus velocidades lineal y angular y los ángulos de la rueda central virtual.

Un nodo de ROS se encarga de almacenar los mensajes de salida robot real, transformarlos en mensajes de entrada para el robot simulado y volver a reproducirlos con la sincronización adecuada. En el anexo B se explica cómo son estos mensajes.

Además, otro nodo almacena información durante la simulación y dibuja la variación en el tiempo de las velocidades lineal y angular y del ángulo virtual frontal de las robots real, simulado y teórico. Adicionalmente, se calculan una serie de valores sobre los tres robots y las relaciones entre sus comportamientos en cada prueba. Se mide la distancia total recorrida en cada trayectoria, además del tiempo que le cuesta recorrerla. El anexo D explica con más detalles cómo se organizan los diferentes nodos.

A partir de esos datos se puede obtener la velocidad lineal media. También se calcula la velocidad angular media. Para ambos casos se utilizan valores absolutos. En el caso de la velocidad lineal no es realmente importante, ya que las pruebas se realizan avanzando, al ser éste el comportamiento más habitual del robot. Sin embargo, para la velocidad angular es vital considerar valores absolutos, ya que de lo contrario en las pruebas que gira aproximadamente lo mismo hacia cada sentido parecería que no ha girado nada.

Otro dato que se considera interesante es la separación final entre las posiciones finales de cada robot, aunque no hay que olvidar que el error de posición se acumula, por lo que también se calculan las separaciones finales por metro recorrido. Aún así, hay que tener en cuenta la posibilidad de que trayectorias muy distintas terminen en puntos cercanos simplemente por azar, por lo que estos valores deben ser tratados con cuidado.

También se computan los errores existentes entre las velocidades y los ángulos de los robots real y simulado. Se calcula su media, su desviación típica y la media de sus valores absolutos. Esta última media informa sobre la magnitud de los errores, mientras que los dos primeros también nos permiten saber algo sobre su distribución.

Por último, se buscan correlaciones entre los errores de velocidad lineal y angular y las propias velocidades lineal y angular. El coeficiente de Pearson mide la correlación lineal y el de Spearman la correlación entre su orden [17]. Se utilizan las correlaciones de la simulación, ya que son conocidas y pueden permitir corregir esos errores. Los valores reales no son conocidos durante la simulación.

Para conocer lo significativas que son esas correlaciones se calcula la probabili-

Rueda	Radio medido (m)	Radio ajustado (m)
Delantera izquierda	0,295	0,294
Delantera derecha	0,298	0,299
Trasera izquierda	0,302	0,300
Trasera derecha	0,306	0,307

Tabla 4.1: Radios de las ruedas ajustados.

dad de que se produzcan bajo la hipótesis nula de que no existe correlación. Se asume que la distribución que ésta sigue es una t de Student con n - 2 grados de libertad, donde n es el tamaño de la muestra [17]. A esta probabilidad se le llama valor p.

4.2. Ajuste del modelo

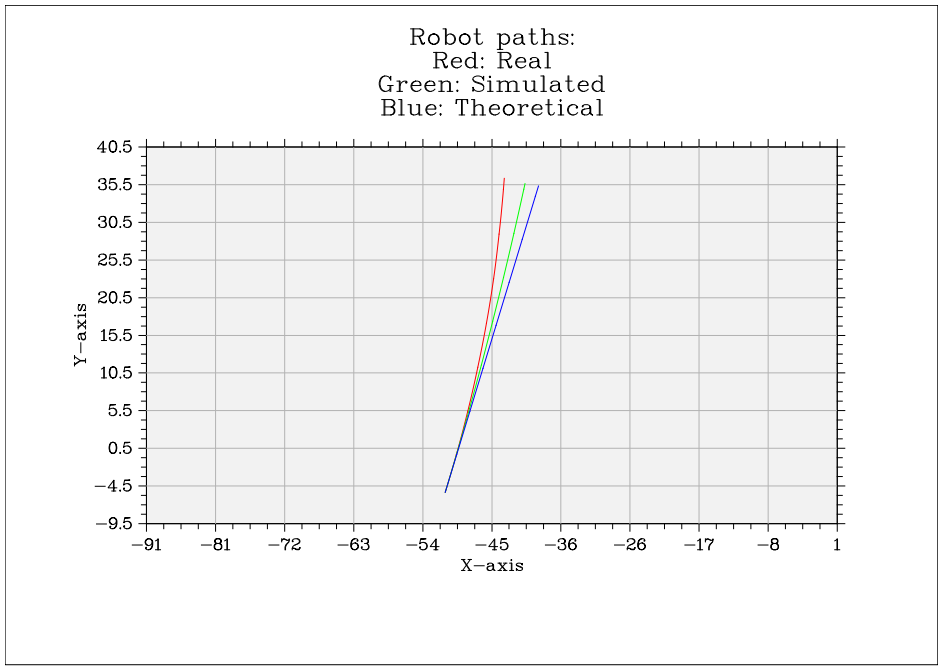
Una vez realizadas las distintas pruebas propuestas en la sección 4.1, cuyos resultados se presentan en el anexo E, se puede pasar a corregir los problemas observados. Los más importantes son el desvío hacia la izquierda debido a las diferencias entre radios, el error en la velocidad lineal en giros hacia la derecha y el deslizamiento en velocidades angulares elevadas.

El objetivo principal es ajustar las velocidades. Las trayectorias van acumulando error y es más difícil utilizarlas para comprobar lo bien que funciona el simulador. Además, un comportamiento más realista de las velocidades implica que las posiciones también mejorarán.

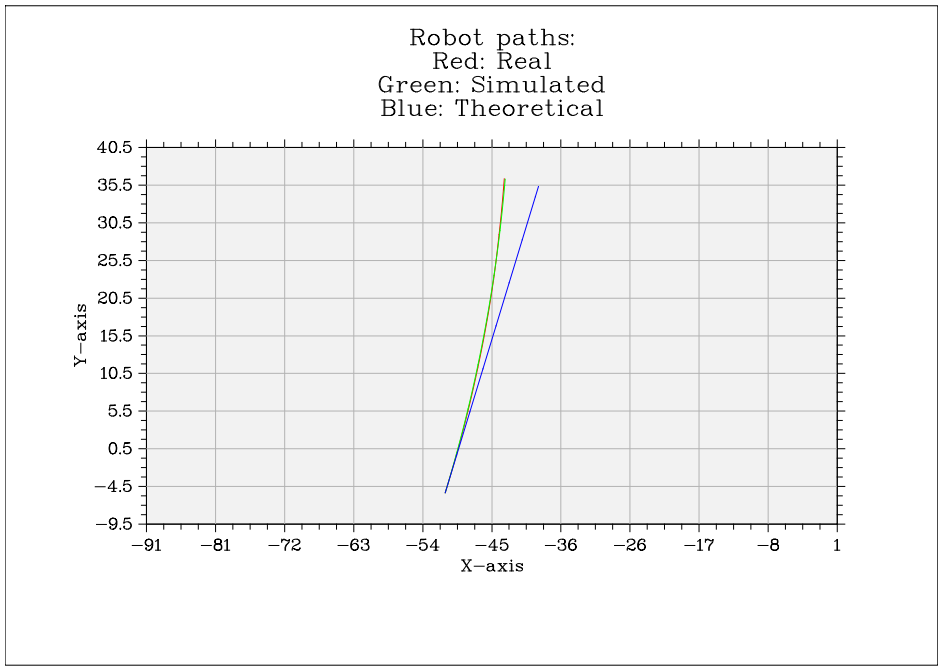
4.2.1. Radio de las ruedas

En primer lugar, se busca imitar la desviación a la izquierda del robot real modificando los radios de las ruedas. Para empezar, se miden las ruedas en la realidad y se introducen esas medidas en el simulador. Acto seguido, se prueba con esos valores y se van modificando hasta llegar a un resultado satisfactorio, sin alejarse mucho de las mediciones. La tabla 4.1 muestra tanto radios medidos como los finales. En la figura 4.1 se muestran las trayectorias tanto utilizando los radios medidos como tras el ajuste. Es destacable que una diferencia de unos pocos milímetros tiene un efecto bastante apreciable en esas trayectorias.

Hay que señalar que estos radios cambian según cómo estén hinchadas las ruedas, la presión y la temperatura, entre otros factores, por lo que este ajuste únicamente será correcto para las circunstancias de esta prueba. Además, hace que



(a) Radios medidos



(b) Radios ajustados

Figura 4.1: Trayectorias utilizadas para el ajuste de los radios, tanto con las medidas reales (a) como las modificadas (b). En azul la trayectoria teórica, en rojo la real y en verde la simulada.

conseguir un ajuste perfecto sea superfluo. A pesar de ello, se consigue reducir el error también para casos cercanos a éste. Adicionalmente, es posible cambiar estos valores modificando un fichero de texto, como se explica en el anexo F.

4.2.2. Velocidad lineal

Un efecto que se ha observado en las pruebas del anexo E es que el simulador presenta un gran error en la velocidad lineal durante los giros a la derecha. Esto se debe a la asimetría introducida por la masa que simula el brazo robótico.

Para solucionarlo se puede colocar en el centro del robot, haciendo que los resultados sean mucho mejores, como se ve en la figura 4.2. La posición de esa masa cambia la distribución de fuerzas sobre las ruedas, que a su vez afecta al rozamiento del simulador, que no está modelado correctamente, como se explica en la sección 4.2.3.

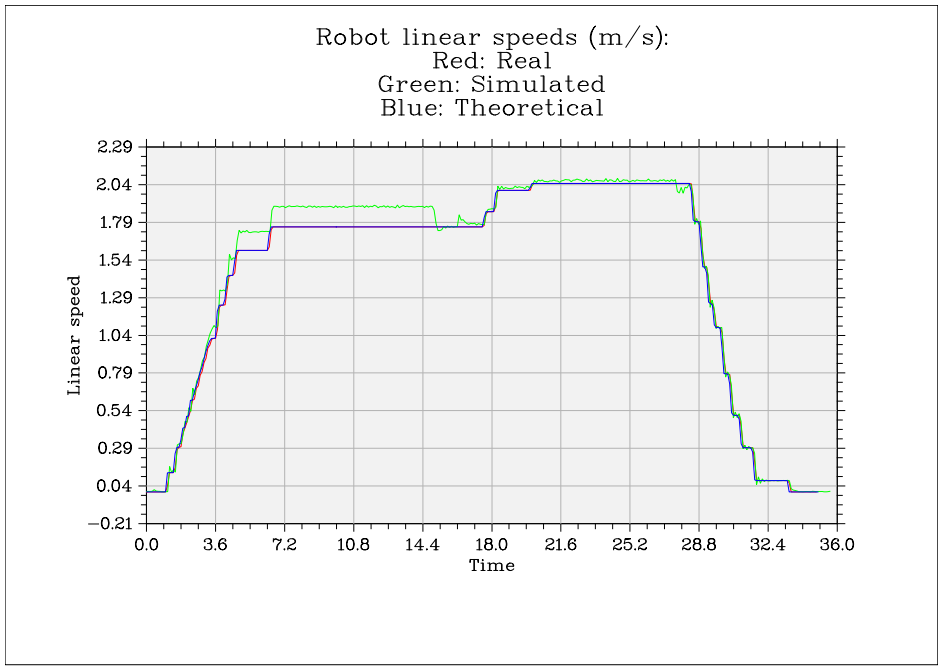
4.2.3. Rozamiento

A partir de una cierta velocidad angular se empieza a producir un deslizamiento muy importante, que impide que ésta alcance los valores teóricos, como se observa en la figura 4.3. Es un fenómeno muy complejo para modelarlo con el rozamiento de Coulomb que permite utilizar el motor de física del simulador. Éste también permite incluir un valor de deslizamiento, pero el comportamiento no es lineal y no proporciona resultados satisfactorios.

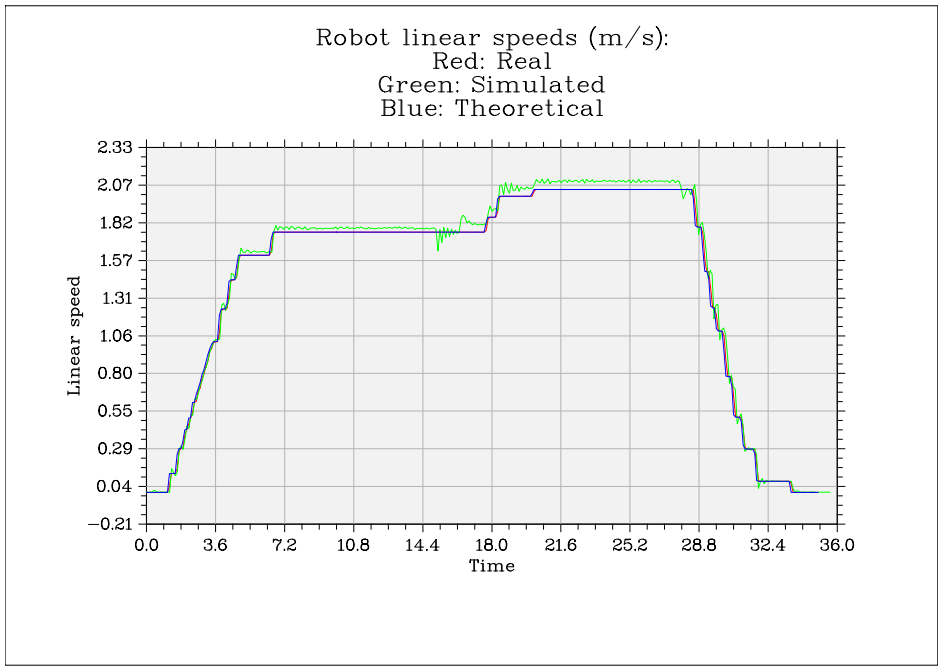
La solución que se plantea es medir en las pruebas realizadas la velocidad angular real en función de la teórica y utilizar ese resultado para modificar la simulación. Dado que a bajas velocidades el deslizamiento no es apreciable y que a altas la velocidad angular parece alcanzar un límite, se definen tres tramos. En el primero la velocidad angular del simulador, w_s, será la teórica, w_t y en el último será ese valor límite, $w_{s\text{límite}}$. En el segundo tramo variará linealmente en función de la velocidad teórica entre los valores de los extremos de los otros dos tramos, de forma que se conserve la continuidad entre todos.

$$w_s = \begin{cases} w_t & si \quad w_t \leq w_{\text{desl}} \\ w_{\text{desl}} + (w_t - w_{\text{desl}}) \dfrac{w_{t\,\text{límite}} - w_{s\,\text{límite}}}{w_{s\,\text{límite}} - w_{\text{desl}}} & si \quad w_{\text{desl}} < w_t < w_{t\,\text{límite}} \\ w_{s\,\text{límite}} & si \quad w_t \geq w_{t\,\text{límite}} \end{cases} \tag{4.1}$$

w_{desl} es la velocidad angular a partir de la cual se deja de considerar lineal,



(a) Brazo desplazado



(b) Brazo central

Figura 4.2: Velocidades lineales para el ajuste de la posición de la masa que simula el brazo robótico. En azul la velocidad teórica, en rojo la real y en verde la simulada.

Figura 4.3: Prueba de giro donde se observan las trayectorias, marcadas como surcos en la grava. Los radios de giro crecen con la velocidad para un ángulo de ruedas constante.

mientras que $w_{t\,\text{límite}}$ es el valor de velocidad teórica a partir del cual se considera que la simulada no aumenta y permanece constante en $w_{s\,\text{límite}}$.

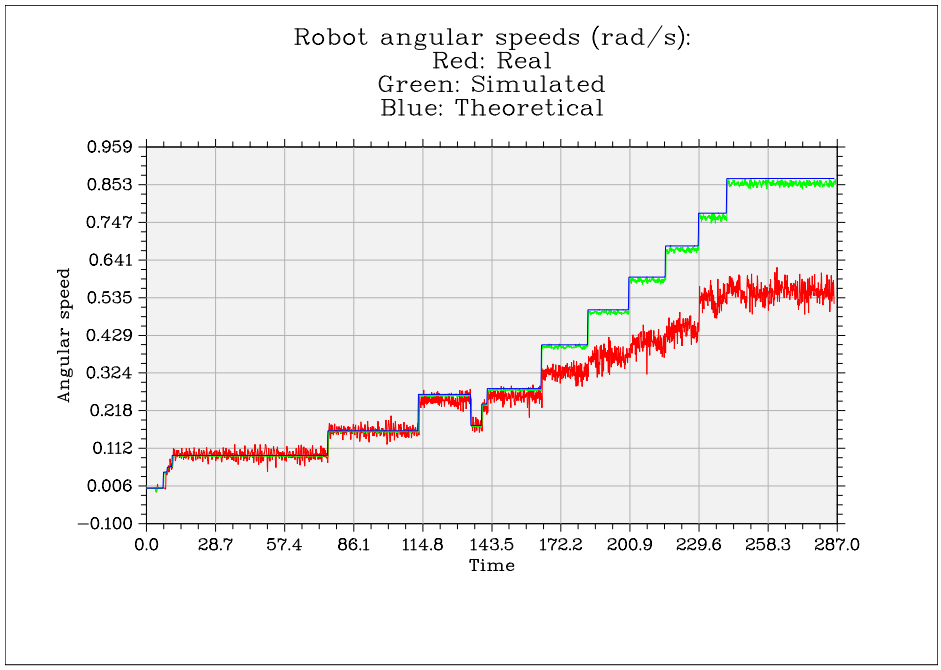
Observando las pruebas de giros en el anexo E se estima que el deslizamiento empieza a ser apreciable a partir de los 0,3 rad/s y que en torno a los 0,55 rad/s no aumenta más la velocidad angular.

Por supuesto, este ajuste únicamente es correcto para las condiciones de esas pruebas. Por ello, se incluye la posibilidad de modificarlo o desactivarlos, según se explica en el anexo F, donde se recoge la configuración de todos los parámetros de ajuste. Además, hay que tener en cuenta que utilizarlo implica que los ángulos de las ruedas del robot simulado no serán correctos.

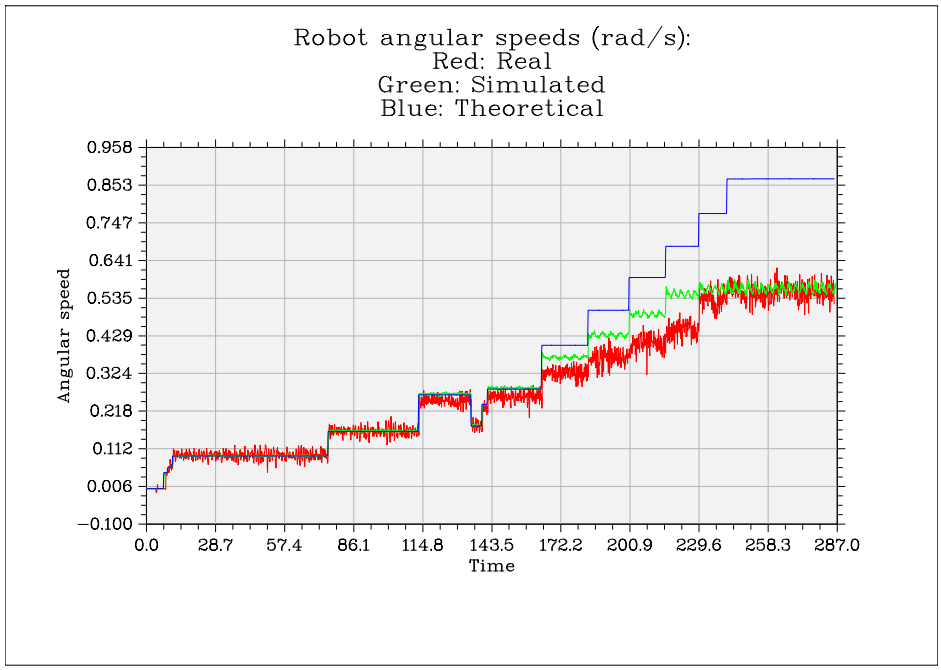
El resultado se presenta en la figura 4.4. La velocidad angular simulada se acerca mucho más a la real, aunque sigue sin ser perfecto. Sin embargo, se puede considerar un compromiso razonable entre el error y la sencillez para modificarlo. Únicamente es necesario introducir dos puntos para que quede definido por completo.

4.3. Evaluación del simulador

Una vez realizadas varias modificaciones en el simulador, se vuelven a realizar varias pruebas para evaluar cuánto ha mejorado. Sus resultados también se muestran en el anexo E.



(a) Giro sin correción



(b) Giro con correción

Figura 4.4: Velocidades angulares sin corrección de deslizamiento (a) y con correción de deslizamiento (b). En azul la velocidad teórica, en rojo la real y en verde la simulada.

El error que se producía en las rectas debido a los radios de las ruedas prácticamente se ha eliminado de la recta con la que se ajustó. La separación final se ha reducido aproximadamente 36 veces. Al probar con otra, se puede observar que se aproxima bastante, pero sí existe un pequeño error. Esto se debe a que la otra recta se probó en condiciones distintas, y cualquier pequeña variación en los tamaños de las ruedas se traduce en errores considerables. De cualquier forma, este error es más de diez veces más pequeño que antes del ajuste, de unos tres milímetros por cada metro recorrido.

Para una trayectoria con curvas a una velocidad elevada los resultados se acercan más que antes de ajustar, reduciéndose el error en un 228%. Sin embargo, siguen sin ser muy precisos, con un error medio de 0,0043 rad/s. Esto se debe a que la corrección de velocidad angular se realiza para unas condiciones en concreto, que no son las de esta trayectoria. Sería necesario reajustar para esta situación o simplemente asumir que para altas velocidades angulares existe un error que debe ser tenido en cuenta.

Al probar con una trayectoria más larga se observa que los errores de posición, acumulados a lo largo del ensayo, son bastante considerables. Sin embargo, las velocidades y los ángulos presentan errores muy reducidos. La principal causa del error en este caso es, de nuevo, la velocidad angular, con una media de 0,0054 rad/s. La figura 4.5 muestra esa trayectoria, pudiéndose observar la diferencia entre el robot real y el simulado. La figura 4.6 muestra las velocidades angulares para esa prueba, donde se puede ver que la simulación se ajusta bien a la realidad. La velocidad lineal y el ángulo de la rueda se presentan en las figuras E.42 y E.44, en el anexo E.

En general, se puede decir que el simulador es bastante realista para velocidades angulares reducidas. En cuanto el fenómeno del deslizamiento empieza a ser relevante los resultados empeoran considerablemente. Las posiciones terminan teniendo errores grandes, pero se debe principalmente a que se acumulan. Las velocidades lineales y los ángulos de la rueda virtual central se reproducen con exactitud, al igual que la velpcidad angular si ésta es reducida.

4.3.1. Estimación del error

Dado que es inevitable la aparición de errores, es interesante saber cuál es su magnitud. A la hora de utilizar un robot, es de vital importancia conocer no sólo la posición en la que cree estar sino además acotar lo precisa que ésta es. Posteriormente, esta información puede ser usada con un filtro de Kalman [18] para mejorar las estimaciones de la posición.

Para utilizar ese filtro la odometría del robot debe incluir una matriz de covarianzas para su posición y orientación. Ésta se incluye en el mensaje de ROS de posición, como se explica en el anexo B. Esta matriz expresa las covarianzas

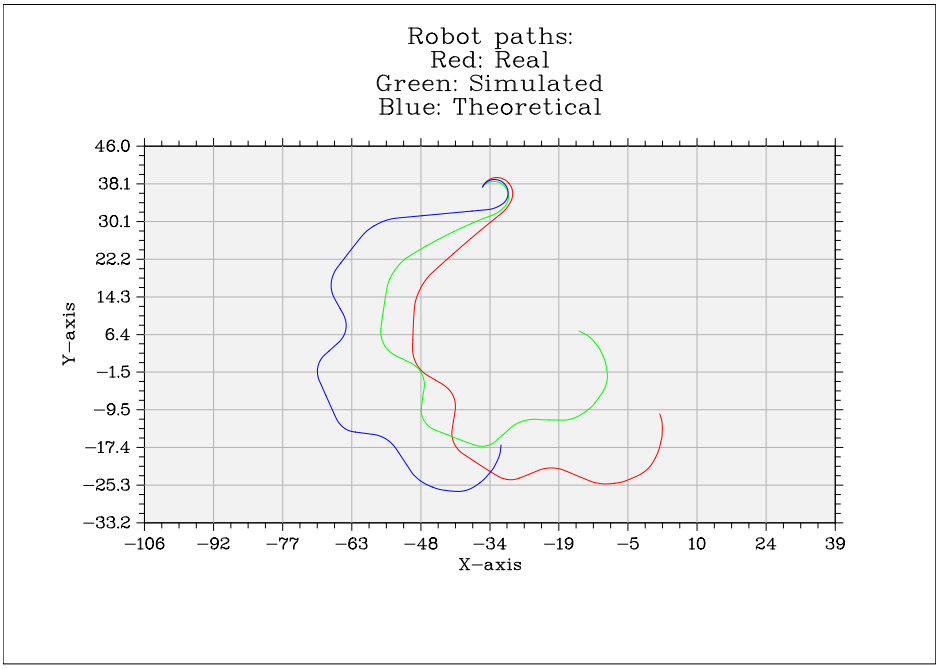


Figura 4.5: Diferentes trayectorias más larga. En azul la trayectoria teórica, en rojo la real y en verde la simulada.

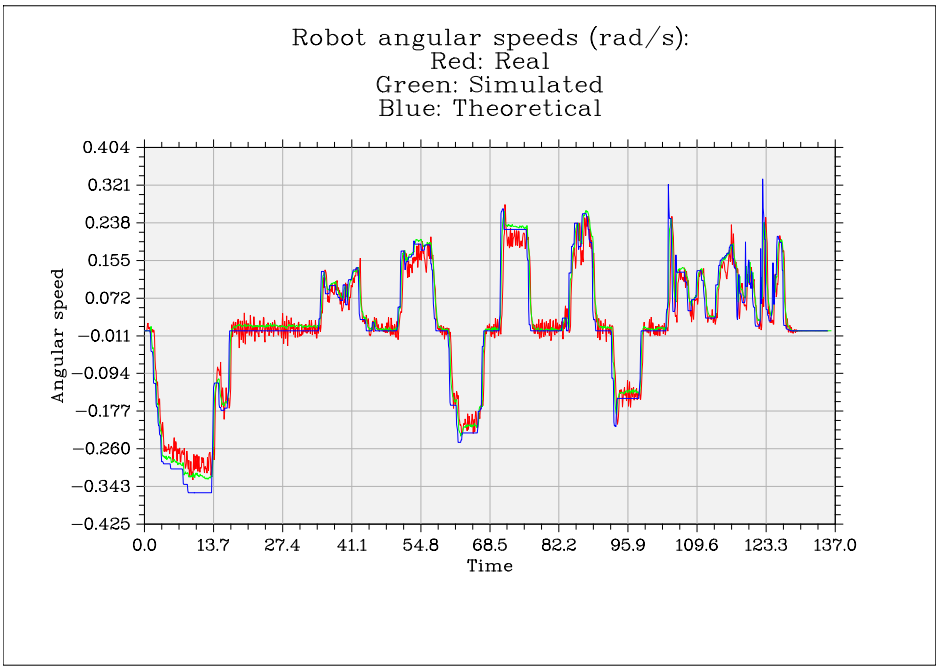


Figura 4.6: Diferentes velocidades angulares de la trayectoria de la figura 4.5.

entre los elementos de un vector. Para el compuesto por las dos coordenadas cartesianas x e y y la orientación θ de un movimiento en dos dimensiones es una matriz simétrica 3×3:

$$\Sigma = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 \end{pmatrix}, \tag{4.2}$$

donde los elementos de la diagonal principal son varianzas y el resto covarianzas.

Con el objetivo de simplificar la obtención de estos valores, se busca su relación con las varianzas de la distancia recorrida, s, y el ángulo de la rueda virtual central, ϕ. Para ello se linealiza utilizando un desarrollo de Taylor de primer orden:

$$\Sigma = J Q J^T. \tag{4.3}$$

Se considera que el error de s aumenta linealmente con la velocidad, mientras que con ϕ se considera constante. Q es la matriz que contiene esas dos varianzas, además de su covarianza:

$$Q = \begin{pmatrix} |s| \cdot \sigma_s^2 & \sigma_{s\phi} \\ \sigma_{s\phi} & \sigma_\phi^2 \end{pmatrix}, \tag{4.4}$$

mientras que J es la matriz jacobiana de la función que relaciona los incrementos de posición y ángulo con s y ϕ. En el anexo G se explica cómo se obtienen sus términos.

A la hora de estimar los errores se utiliza una prueba real del robot y se compara con el comportamiento de la simulación. Se escoge el peor caso posible, una trayectoria donde la velocidad del robot sea elevada. Además, dado que el error de deslizamiento aumenta linealmente con la velocidad angular [19], se utiliza el ángulo máximo de las ruedas. Tampoco se usa el ajuste del deslizamiento con el ángulo de las ruedas. Los resultados se presentan en la tabla 4.2.

Estos valores se introducen en el simulador tal y como se explica en el anexo F. Además, también se explica cómo modificarlos en el caso de que sea necesario.

4.4. Resumen de resultados

A continuación, se presenta un breve resumen de los resultados obtenidos tanto en el análisis de sensibilidad, presentado en la sección 3.4, como del ajuste realizado en el presente capítulo.

Elemento	Valor estimado
σ_s^2	0,0001373 (m^2)
σ_ϕ^2	0,0001985 (rad^2)
$\sigma_{s\phi}$	-0,00002115 ($m \cdot rad$)

Tabla 4.2: Estimación de la matriz de covarianzas Q.

Los dos factores que más error introducen son:

- La variación en el tamaño de las ruedas. En el anexo C se ha comprobado que pequeñas variaciones en ese tamaño producen grandes cambios en el comportamiento de la simulación. Además, en la sección 2.2.1 se explica que este tamaño no es constante, siendo distinto para diferentes condiciones.
- El deslizamiento lateral del robot. A velocidades angulares elevadas, por encima de aproximadamente 0,3 rad/s, se produce un importante efecto de deslizamiento que puede dar lugar a errores muy importantes, como se explica en la sección 4.2.3. Además, al depender tanto del estado del terreno como de las condiciones atmosféricas no puede ajustarse el modelo sin más, ya que cambia para cada caso.

Se pueden reducir estos errores modificando el ajuste para cada caso concreto, tal y como se explica en el anexo F. Además, en la sección 4.3.1 se propone un modelo de error gaussiano para el simulador, que permite acotarlo.

CAPÍTULO 5

MODELADO VISUAL DEL ROBOT

Este capítulo versa sobre el aspecto visual del robot y cómo reflejarlo en el simulador Gazebo, mediante el uso de una cámara RGB-D. En primer lugar se describen las herramientas utilizadas, tanto la cámara como el *software*. Después se explica cómo se alinean varias capturas de esa cámara, cuya información se almacena en las llamadas nubes de puntos. Por último, se proponen varios métodos para la reconstrucción de superficies a partir de una serie de nubes de puntos alineadas.

Utilizar este tipos de métodos en vez de un programa de CAD tiene como propósito de facilitar la creación de otros modelos tridimensionales. En especial, se podrían crear escenarios más realistas para el simulador en menos tiempo.

5.1. Herramientas utilizadas

En esta sección se explica brevemente cómo es la cámara utilizada y las dos librerías más importantes que se han usado: PCL y OpenCV.

5.1.1. Cámara

Para la captura de las imágenes se utiliza una cámara RGB-D. En concreto, se utiliza un dispositivo Kinect, desarrollado inicialmente por Microsoft para su videoconsola Xbox 360.

Las siglas RGB significan *Red Green Blue* y se refieren a una cámara digital normal. Ésta almacena las imágenes como matrices bidimensionales de píxeles. Éstos contienen tres campos con los colores rojo, verde y azul.

La D significa profundidad, del inglés *depth*, debido al sensor de profundidad con el que estas cámaras cuentan. Un proceso de triangulación es utilizado para

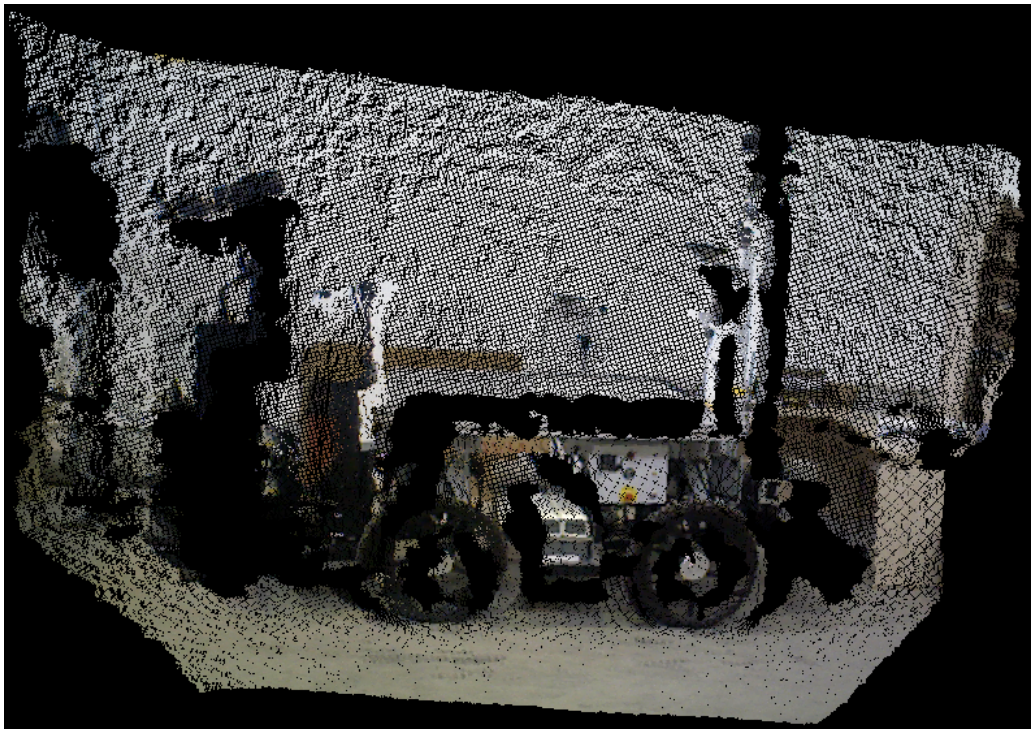


Figura 5.1: Ejemplo de nube de puntos que muestra al robot.

obtenerla [20]. Hay que tener en cuenta que la resolución disminuye con el cuadrado de la distancia a la cámara. Pasa de aproximadamente 2 mm a un metro a 2,5 cm a tres metros y 7 cm a cinco metros [21]. Esto limita el rango de visión, ya que, a pesar de que la cámara es capaz de detectar objetos hasta unos seis metros, la precisión de esas medidas es muy reducida. En la práctica se recomienda limitar la distancia a unos tres metros, dependiendo de la aplicación. También existe un límite inferior en torno a 60 cm.

Un método para almacenar la información combinada de color y profundidad es mediante una nube de puntos. Cada píxel se coloca en un espacio cartesiano tridimensional utilizando su profundidad y su posición en la imagen RGB bidimensional. Al visualizarse, el conjunto de puntos recuerda a una nube, como se puede ver en la figura 5.1.

Con la ayuda de OpenNI, un *driver* de código abierto, se pueden capturar nubes de puntos con el dispositivo Kinect para su posterior uso.

5.1.2. PCL

PCL, *Point Cloud Library*, es una herramienta que se puede utilizar el procesado de nubes de puntos [22] [23]. Originalmente surgió como un paquete de

ROS. De hecho, es posible añadir uno o varios sensores Kinect al robot y que éste reciba y procese su información dentro de uno o varios nodos de ROS.

Incluye una serie de funciones para su alineación y modificación. Además, tiene implementados varios algoritmos para la reconstrucción de superficies a partir de nubes. Posteriormente, esas superficies pueden ser guardadas en un formato compatible con Gazebo como Collada. Para ello se utiliza un programa para la modificación de mallas, como MeshLab.

5.1.3. OpenCV

Para el procesado de imágenes en dos dimensiones se puede usar OpenCV, *Open Source Computer Vision* [24]. También está integrado en ROS. Entre muchas otras funciones, permite la búsqueda de emparejamientos entre dos imágenes, algo que se utiliza en el proceso de alineamiento de nubes de puntos.

5.2. Alineación de nubes de puntos

Es necesario utilizar varias nubes de puntos tomadas desde distintos puntos para lograr una reconstrucción completa de una escena o de un objeto. Por ello, conocer la transformación que permita pasar todas ellas al mismo sistema de referencia es imprescindible. Este proceso se conoce como registro o alineación de nubes de puntos.

En primer lugar, se buscan correspondencias entre las imágenes bidimensionales. Para ello se utiliza el extractor de *keypoints*, o puntos clave, SURF (*Speeded Up Robust Features*) implementado en OpenCV [25]. Estos puntos se obtienen de la imagen transformada a escala de grises. También se computa un descriptor para cada uno.

Se buscan los *keypoints* correspondientes entre las imágenes utilizando un *Flann Based Matcher*, emparejador basado en Flann (Fast Approximate Nearest Neighbors) [26]. Es otro algoritmo que ya está implementado en OpenCV. La figura 5.2 muestra este proceso para dos imágenes. Se presentan únicamente algunos de ellos para facilitar su visualización.

Después es necesario rechazar los emparejamientos erróneos, mediante el uso de RANSAC (*Random Sample Consensus*) [27]. De nuevo, se utiliza la versión disponible en OpenCV.

Una vez eliminados esos emparejamientos erróneos se transforman los restantes a tres dimensiones en la nube de puntos y se vuelve a aplicar RANSAC, con dos objetivos. El primero es rechazar algún emparejamiento erróneo que todavía pueda quedar. El segundo es obtener una primera aproximación de la matriz

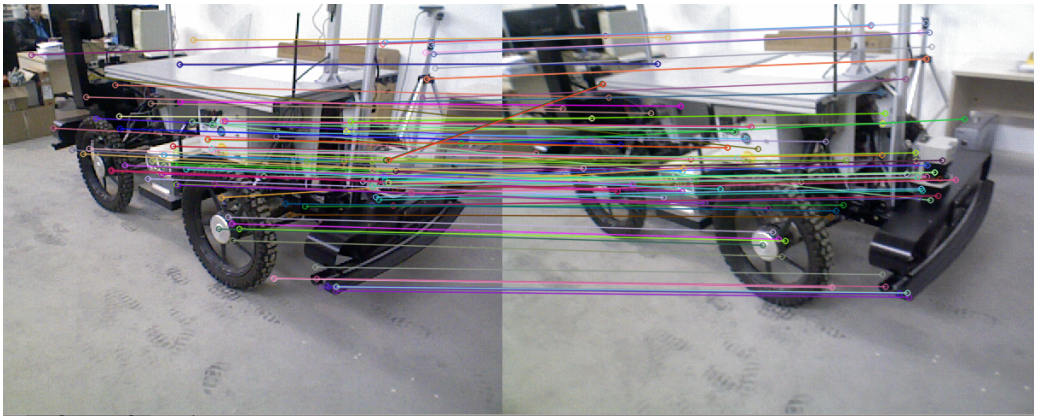


Figura 5.2: Emparejamientos entre dos imágenes utilizando el *Flann Based Matcher*.

de transformación entre las nubes. Al estar trabajando con nubes se utilizan los métodos de PCL a partir de este punto.

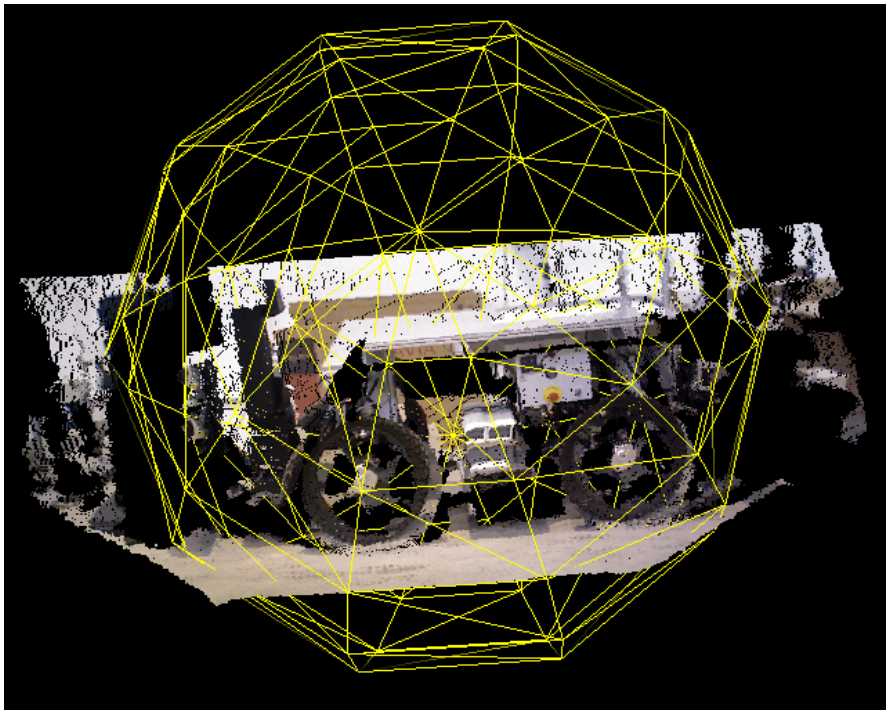
Tras ese alineamiento preliminar se aplica ICP, *Iterative Closest Point* [28]. Este algoritmo busca minimizar los errores entre dos nubes que ya estén aproximadamente alineadas. Una vez realizado se obtiene una matriz de transformación que permite pasar la segunda nube al sistema de referencia de la primera y se da por concluido el proceso de registro.

Las sucesivas nubes se alinean con la anterior y se les aplica esa transformación y todas las anteriores para que el sistema de referencia de todas ellas sea el mismo, correspondiente con el de la primera nube.

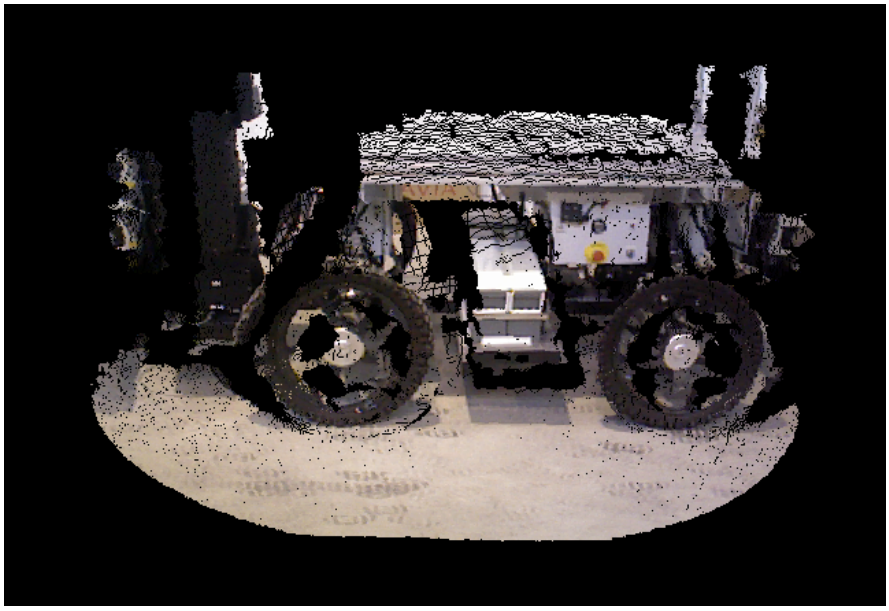
5.3. Reconstrucción de superficies

Antes de proceder a reconstruir las superficies es necesario eliminar la parte de las nubes que no se corresponden con el objeto que se quiere utilizar. Para ello, se asume que estará situado encima del suelo y suficientemente alejado de otros objetos. Se proporciona una herramienta para escoger la porción de espacio donde está el objeto y se eliminan los puntos exteriores, como se muestra en la figura 5.3. Para simplificar la elección del espacio se escoge un punto central y una distancia máxima a ese punto, formando una esfera.

Después se busca el mayor plano de la nube restante y se elimina. Se asume que este plano es el suelo. Por supuesto, este método presenta importantes limitaciones, pero es útil como una primera aproximación que permite continuar con el proceso. Se pide confirmación antes de eliminarlo, ya que no tendría sentido utilizarlo para escenarios, donde en principio es interesante conservar las nubes



(a) Colocación de la esfera



(b) Resultado

Figura 5.3: Recorte de nubes de puntos. En (a) se muestra la nube donde se puede modificar la esfera amarilla y en (b) el resultado.

Figura 5.4: Nube de puntos utilizada para los ejemplos de reconstrucción de superficies.

enteras.

Es posible que existan puntos espurios, que deben ser encontrados y eliminados. Se utiliza un algoritmo que elimina los puntos que estén alejados más de una desviación típica de la distancia media a sus vecinos más cercanos [29]. También se utiliza un *Voxel Grid* para reducir el tamaño de las nubes, con el fin de acelerar el proceso. Este método llena de cuboides el espacio y dentro de cada uno aproxima los puntos por su centroide.

Antes de utilizar los algoritmos para reconstruir las superficies es necesario estimar sus vectores normales. Para ello se utiliza un método que modifica esos vectores y consigue superficies más lisas y uniformes, aproximando por polinomios con mínimos cuadrados móviles (*Moving Least Squares*, MLS) [30]. Es importante reseñar que en algunos casos este método provoca superficies ligeramente onduladas, algo que debe ser tenido en cuenta cuando se usa.

Se proponen varias alternativas para la reconstrucción de superficies. La primera es *Greedy Projection Triangulation* [31]. Un algoritmo *greedy*, que se puede traducir como voraz o ávido, sigue una heurística para buscar soluciones óptimas locales. De esta manera se puede conseguir una buena solución global, aunque no se garantiza. En este caso lo que se hace es crear un mallado triangular.

El segundo método es la reconstrucción de superficies por Poisson (*Poisson Surface Reconstruction*) [32]. Busca una solución global, lo que lo hace menos

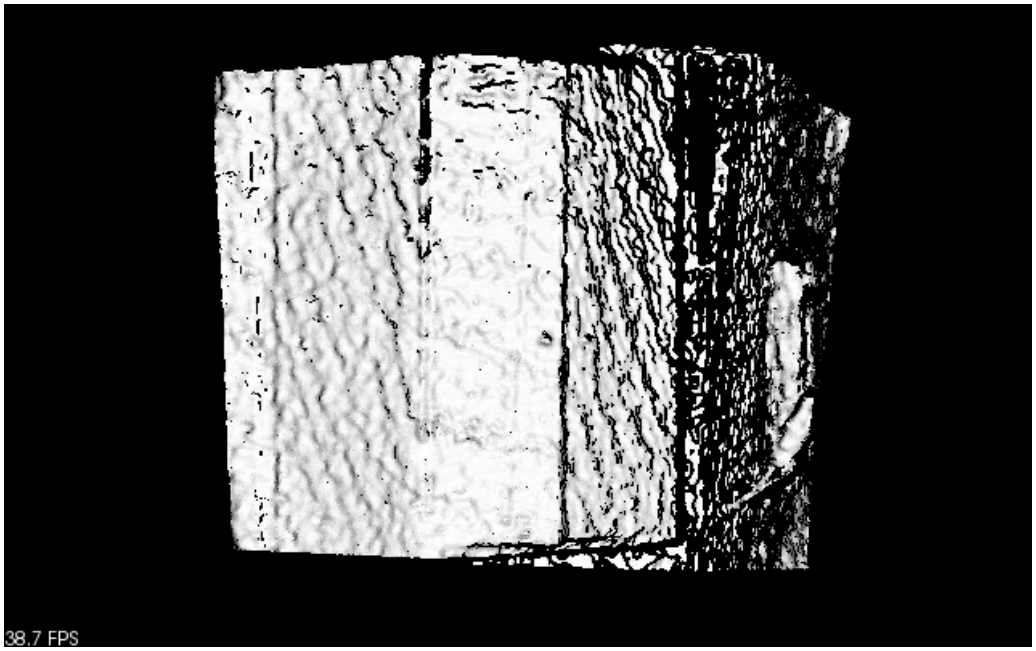


Figura 5.5: Ejemplo de reconstrucción de una superficie mediante *Greedy Projection Triangulation*.

sensible al ruido. Sin embargo, siempre cierra la superficies, lo que es un problema cuando no se está intentando reconstruir un objeto cerrado. Si el objeto de la nube tiene varias partes las intenta unir de algun manera, generando superficies que no tienen ningún sentido.

Otra posibilidad es utilizar *Grid Projection* [33]. Los resultados que produce son parecidos a los del primer método, pero es mucho más lento, pasando de unos pocos segundos a varias horas, dependiendo del tamaño de la nube de puntos utilizada.

Por último, se aplica a los resultados un suavizado de malla Laplaciano [34], con el objetivo de reducir la rugosidad de las superficies generadas.

Se aplican estos métodos a una pequeña parte de un escenario, mostrado en la nube de la figura 5.4. La figura 5.5 muestra un ejemplo del primer método presentado, la figura 5.6 del segundo y la figura 5.7 del tercero.

Sería posible realizar varias modificaciones para permitir la reconstrucción de superficies a partir de un número mayor de nubes de puntos, que permitiría crear modelos tridimensionales de escenarios más grandes y del robot.

Se puede utilizar cerrado de bucles basado en SLAM, *Simultaneous Localization and Mapping* [35], fijando una nube de puntos como referencia y transformando simultáneamente todas las demás a esa referencia [36] [37].

En [38] se aplica para mejorar el alineamiento de las nubes. En este proyecto

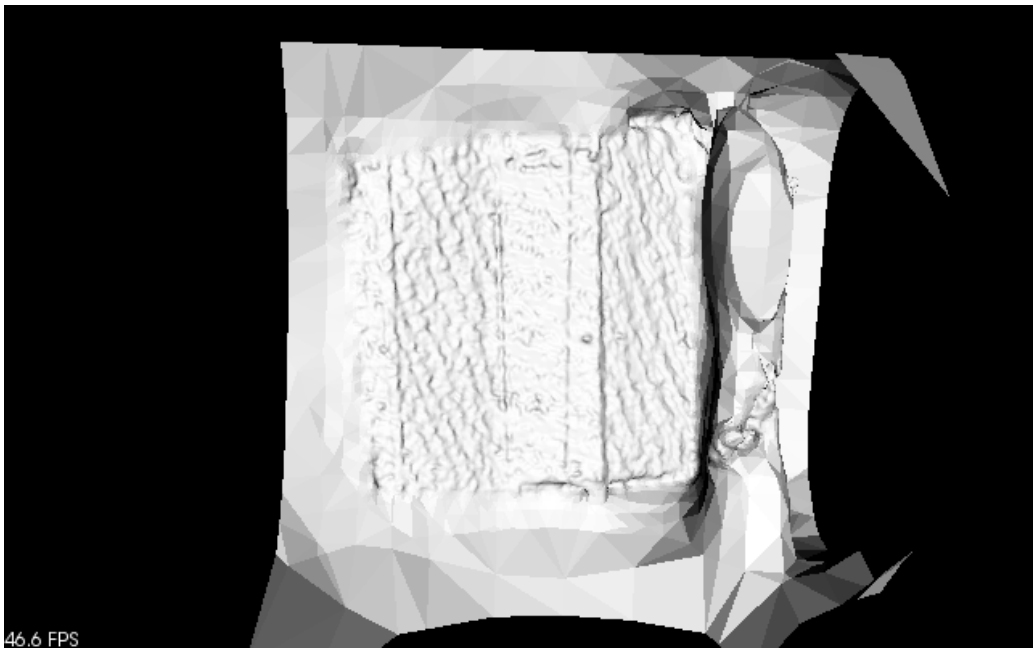


Figura 5.6: Ejemplo de reconstrucción de una superficie mediante *Poisson Surface Reconstruction*.

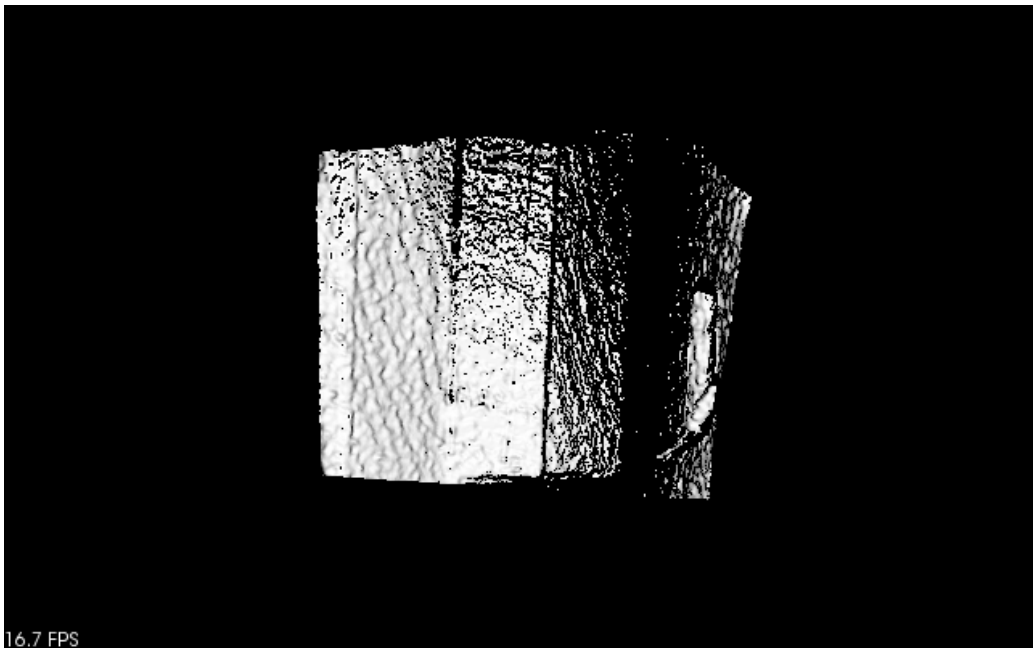


Figura 5.7: Ejemplo de reconstrucción de una superficie mediante *Grid Projection*.

simplemente se alinean consecutivamente, acumulando error e impidiendo usar más de un número reducido. Este método permite tener en cuenta bucles cerrados de nubes en su conjunto, lo que permite un alineamiento correcto de cientos de ellas. También es posible extenderlo para permitir escenarios aún mayores [39].

Otra posibilidad de mejora sería probar otros métodos de reconstrucción de superficies, como *Marching Cubes* [40] [41], que podrían dar lugar a resultados más cercanos a la realidad.

5.4. Conclusiones

Los resultados obtenidos en este capítulo no tienen la calidad suficiente como para poder ser utilizados dentro de Gazebo. Se ha logrado una aproximación a superficies sencillas, pero al aplicarlo a nubes de puntos más complejas, como el robot de la figura 5.3b, no se logra una buena superficie.

Por lo tanto, y a pesar de que el trabajo realizado con la cámara Kinect ha supuesto aproximadamente una cuarta parte de las horas dedicadas a este proyecto, se ha decidido crear una representación simplificada del robot utilizando las herramientas proporcionadas por Gazebo. Ésta se muestra en la figura 5.8.

Como se ha mencionado en la sección anterior, el uso de algoritmos que permitan el alineamiento de cientos de nubes de puntos y la utilización de nuevos métodos de reconstrucción de superficies podrían permitir, en un futuro, el uso de un sensor RGB-D para la recreación de objetos tridimensionales en el simulador Gazebo.

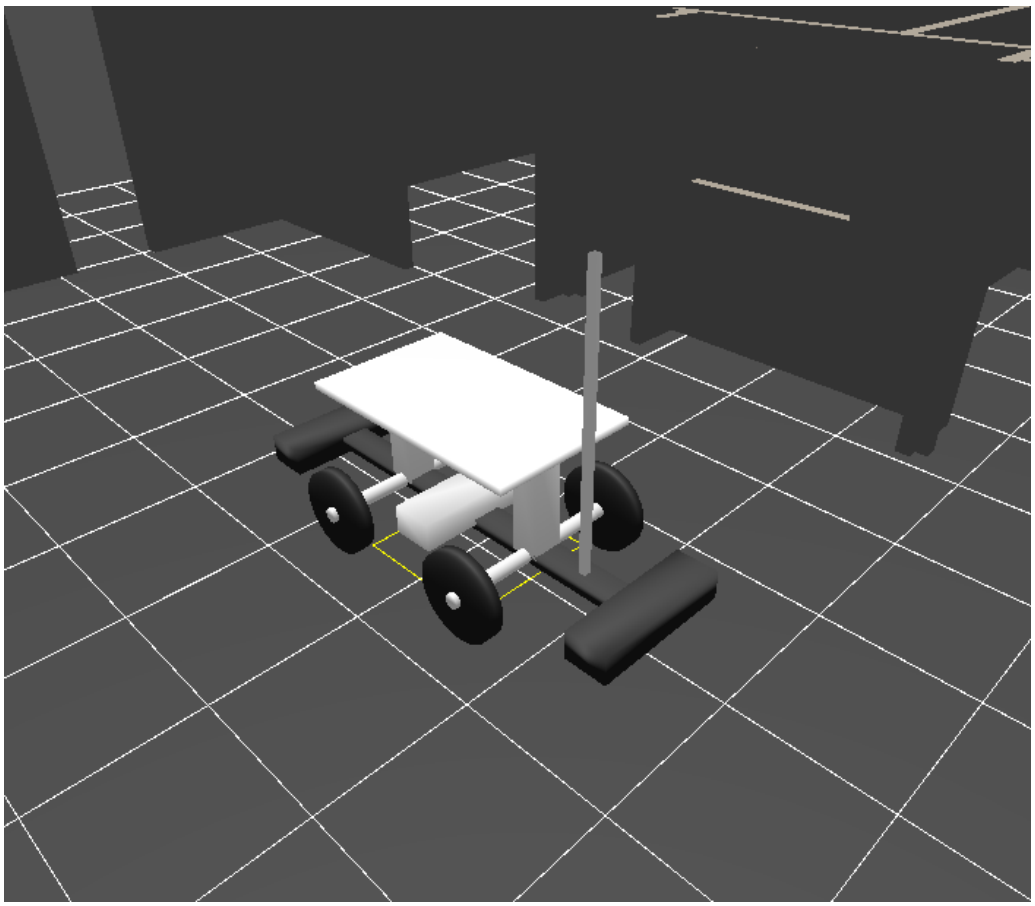


Figura 5.8: Representación simplificada del robot en el simulador Gazebo.

CAPÍTULO 6

CONCLUSIONES Y TRABAJO FUTURO

En este capítulo se presentan algunas conclusiones sobre el trabajo realizado a lo largo de este proyecto. Además, se proponen una serie de posibles mejoras que se podrían llevar a cabo en el futuro.

6.1. Conclusiones

Se ha creado un modelo para el robot Robucar del Grupo de Robótica, Percepción y Tiempo Real y posteriormente se ha implementado en el simulador tridimensional Gazebo. Además, se ha evaluado cómo se ajusta a la realidad y se ha mejorado el modelo.

Las pruebas realizadas indican que su comportamiento se acerca mucho al real para velocidades reducidas. Sin embargo, el deslizamiento lateral de las ruedas se convierte en una factor determinante para velocidades mayores. Dada la dificultad que entraña su modelado, es inevitable la aparición de importantes errores. Además, cambios en las características y condiciones del terreno llevan a variaciones notables. Se ha introducido una herramienta para simular este deslizamiento en casos específicos, ajustando manualmente los parámetros tras una prueba real.

El otro factor más influyente es el radio de las ruedas, cuyos neumáticos cambian de tamaño con el tiempo. Sin embargo, este fenómeno es más sencillo de tratar, ya que estos radios se pueden medir directamente sobre el robot.

El simulador responde exactamente a las mismas órdenes que el robot real, lo que permite utilizarlo con los mismos programas. De estar forma se pueden poner a prueba con más rapidez, seguridad y comodidad. Además, cuando se utilice junto con algoritmos de navegación la precisión del modelo no sería tan importante, dado que se corregiría el comportamiento en función de la información recibida. Por ello, también se podría plantear su uso para todo

el rango de velocidades del robot, incluso aquéllas en las que el deslizamiento presenta una mayor relevancia.

También se considera el error de la simulación respecto a la realidad, con el objetivo de poder utilizar un filtro de Kalman, una herramienta muy común para estimar la posición en robótica.

Se incluye un fichero de configuración que permite cambiar algunos de los parámetros más importantes, para facilitar su posterior modificación en caso de que sea necesario.

Por último, se ha estudiado la posiblidad de utilizar un sensor Kinect para crear un modelo tridimensional del robot y de su entorno. Se ha logrado recrear algún escenario sencillo. Sin embargo, los resultados obtenidos no son lo suficientemente buenos como para ser usados en Gazebo. Por lo tanto, se ha creado un modelo visual para el simulador utilizando las herramientas proporcionadas por éste.

6.2. Trabajo futuro

Sólo se ha incluido en el simulador la posibilidad de utilizar el robot en modo coche. La razón es una avería del robot que impide utilizar los otros dos. Al no poder hacer pruebas para los otros dos modos se ha considerado que su implementación no era prioritaria. Además, el modo más importante es el coche. Sin embargo, añadir esa posibilidad sería interesante, especialmente cuando se repare el robot.

El principal problema encontrado a lo largo del proyecto es la importancia del deslizamiento a velocidades elevadas. Desarrollar un modelo que lo imite con más precisión mejoraría los resultados a partir de velocidades angulares superiores a 0,3 rad/s, aproximadamente. Sin embargo el rozamiento es un fenómeno muy complejo y el simulador Gazebo presenta limitaciones a la hora de representarlo. Además, el robot puede operar sobre distintas superficies y el modelo tendría que ser distinto para cada una. Teniendo en cuenta que Gazebo es una herramienta muy reciente y en desarrollo, se puede esperar a que se incluyan más herramientas para simular el rozamiento en superficies como césped o grava, como está planeado, e introducirlas entonces.

Para el modelo de error se ha buscado el caso más desfavorable y se ha aplicado en todos. Es preferible sobreestimar el error y asegurarse de que el robot quede dentro que hacerlo demasiado pequeño y que la posición en la que esté realmente quede fuera de las posibles para ese error. Sin embargo, es una aproximación bastante conservadora. Se podría plantear un modelo más complejo que tenga en cuenta sus causas en cada caso y se ajuste más, asegurándose en todo momento de que no se hace demasiado pequeño.

También se podrían añadir varios sensores al robot simulado que están presentes en el real. Por ejemplo, existen *plugins* en Gazebo para los láser, el IMU, el GPS y la cámara térmica [42] [43]. Incluso se podría plantear incluir el brazo robótico y su control, bien adaptando otro como el del robot PR2 o creando uno nuevo.

En lo relativo al sensor Kinect, se puede continuar con el trabajo mejorando la alineación de un gran número de nubes mediante cerrado de bucles. También se pueden añadir algoritmos más avanzados para la elección del subconjunto de la nube de puntos que interese, como utilizando alguno de los métodos de segmentación de objetos incluidos en PCL. El uso de otros algoritmos de reconstrucción de superficies también podría mejorar los resultados.

Anexos

ANEXO A

FOTOGRAFÍAS DEL ROBOT

En este anexo se incluyen una serie de fotografías del robot sobre el que trata el proyecto, con el objetivo de poder conocer su aspecto.

Se presenta una visión de conjunto en las primeras imágenes, en las figuras A.1 y A.2. Después se incluyen algunos detalles. Los sensores láser situados tanto delante como detrás del robot se muestran en las figuras A.3 y A.4. El brazo robótico se puede ver en la figura A.5. La figura A.6 incluye una imagen de la geometría de dirección de Ackermann. En la figura A.7 se observan los sensores situados en el poste de la parte trasera, la cámara térmica y el GPS. Por último, los ordenadores montados a bordo del robot se muestran en la figura A.8.

Figura A.1: Fotografía lateral-trasera del robot.

Figura A.2: Fotografía lateral del robot.

Figura A.3: Fotografía delantera del robot.

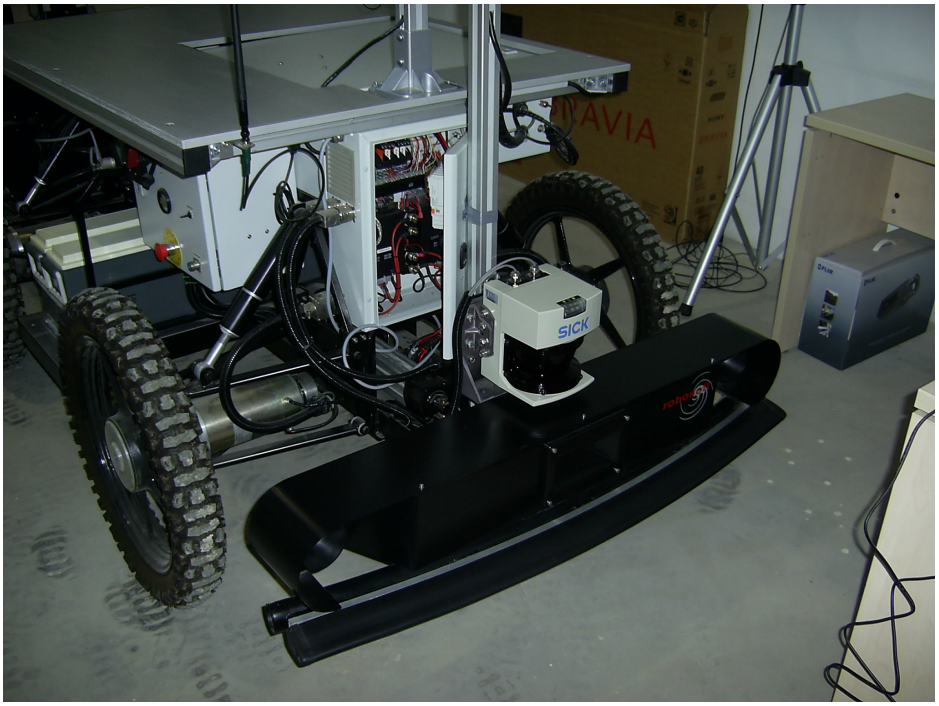


Figura A.4: Fotografía trasera del robot.

Figura A.5: Fotografía del brazo robótico.

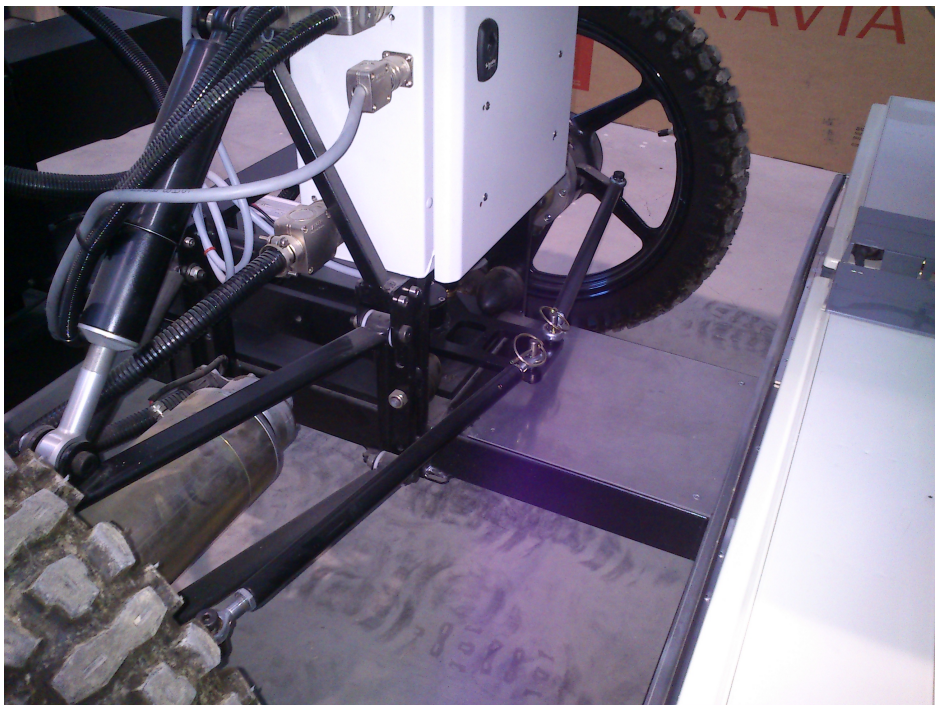


Figura A.6: Fotografía de la geometría de giro de Ackermann.

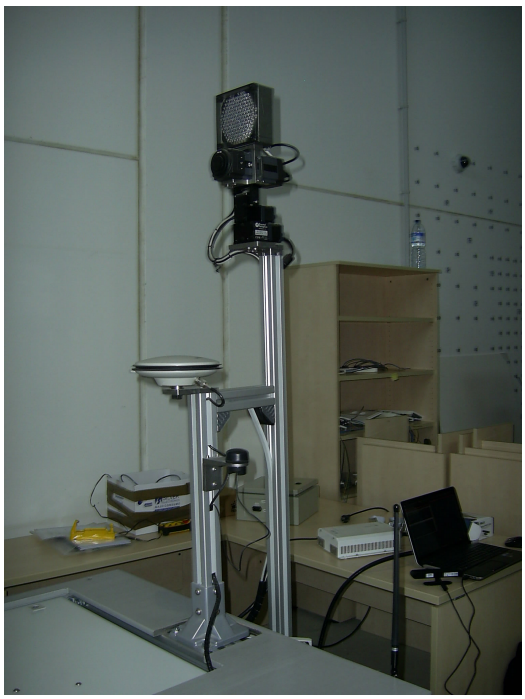


Figura A.7: Fotografía de la cámara térmica y del sensor GPS.

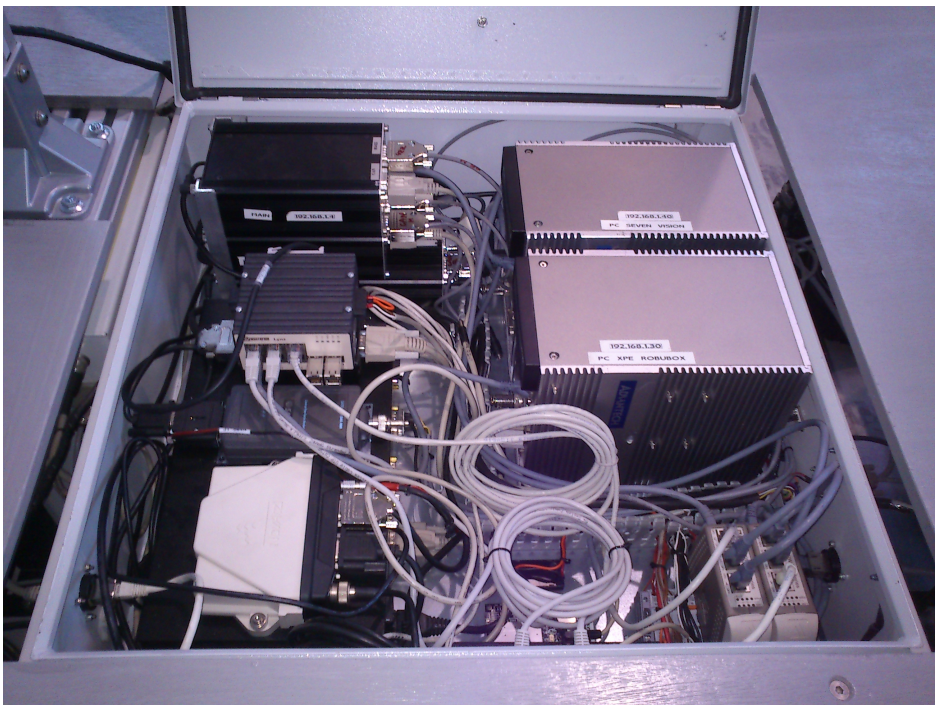


Figura A.8: Fotografía de los ordenadores situados a bordo del robot.

ANEXO B

MENSAJES DE ROS

Este anexo muestra los mensajes de ROS utilizados tanto para enviar las órdenes al robot como para que éste informe sobre su estado. Saber cómo son estos mensajes es imprescindible para poder interactuar con el simulador, y se incluyen para facilitar su consulta. Además, se ofrece una breve explicación de cada uno.

Se han utilizado los mismos que usa el robot real, con el objetivo de hacer más sencillo el uso del simulador.

La primera línea de cada mensaje indica tanto el paquete de ROS como el propio nombre del mensaje. Las demás son los distintos campos del mensaje, incluyendo en primer lugar el tipo y en segundo lugar el nombre.

El primer mensaje, *CarLikeCmd*, es el utilizado para enviar las órdenes al robot en el modo coche. Incluye información sobre la acción que debe realizar, *action*, la velocidad lineal que se le solicita en metros por segundo, *target_speed*, y el ángulo requerido a la rueda virtual central en radianes, *target_steering*. En qué consiste esa rueda virtual se explica en la sección 2.3.1.

```
rosbucar/CarLikeCmd:
 uint8 action
 float32 target_speed
 float32 target_steering
```

El siguiente mensaje, *CarLikeInfo*, es uno de los mensajes que utiliza el robot para proporcionar información. Se incluye el estado del robot, *status*, la velocidad lineal actual en metros por segundo, *current_speed* y el ángulo de la rueda virtual central en radianes, *current_steering*. También se incluyen las órdenes que ha recibido el robot del mensaje anterior. Es muy importante tener en cuenta que, en caso de que el deslizamiento sea importante, calcular la velocidad angular a partir del ángulo de la rueda es muy inexacto.

```
rosbucar/CarLikeInfo:
 Header header
 uint8 status
 float32 current_speed
 float32 current_steering
 float32 target_speed
 float32 target_steering
```

Cuando se incluye información sobre el tiempo en el que se crea el mensaje se incluye como uno de sus campos un *Header*. Este mensaje incluye el momento en el que se crea el mensaje en *stamp*.

```
std_msgs/Header:
 uint32 seq
 time stamp
 string frame_id
```

Una manera alternativa de enviar y de recibir información es utilizando el mensaje *AckermannDriveStamped*, más utilizado por la comunidad de ROS.

```
ackermann_msgs/AckermannDriveStamped:
 Header          header
 AckermannDrive  drive
```

AckermannDriveStamped contiene el campo *drive*, de tipo *AckermannDrive*. Éste se puede utilizar tanto para enviar órdenes como para recibirlas, según en qué *topic* se publique. *steering_angle* es el ángulo de la rueda virtual en radianes y *steering_angle_velocity* es la velocidad a la que cambia en radianes por segundo. La velocidad lineal en metros por segundo es *speed*, la aceleración en m/s^2 es *acceleration* y la derivada de la aceleración en m/s^3 es *jerk*. Se requiere especificar como mínimo *steering_angle* y *speed*. Los demás campos pueden tomar un valor nulo en caso de que no se conozca su valor o no se desee especificarlos.

```
ackermann_msgs/AckermannDrive:
 float32 steering_angle
 float32 steering_angle_velocity
 float32 speed
 float32 acceleration
 float32 jerk
```

La odometría del robot se envía utilizando *Odometry*. Contiene un *Header*, el nombre del sólido del robot que se toma como referencia en *child_frame_id* e información sobre la posición y velocidad del robot en *pose* y *twist*.

```
nav_msgs/Odometry:
 std_msgs/Header header
 string child_frame_id
 geometry_msgs/PoseWithCovariance pose
 geometry_msgs/TwistWithCovariance twist
```

El mensaje *PoseWithCovariance* incluye la posición y orientación del robot en *pose* y la matriz de covarianzas de su modelo gaussiano de error en *covariance*. El mensaje está diseñado para un robot que se mueva en un espacio tridimensional, así que la matriz es 6×6 para las tres coordenadas cartesianas y los tres ángulos de Euler. En un movimiento bidimensional únicamente hay tres grados de libertad, por lo que la matriz se reduce a 3×3.

```
geometry_msgs/PoseWithCovariance:
 geometry_msgs/Pose pose
 float64[36] covariance
```

Hay que destacar que también existe el mensaje *PoseWithCovarianceStamped*, que además de los campos propios de *PoseWithCovariance* también incluye un *Header*.

La posición y la orientación del robot se agrupan en el mensaje *Pose*.

```
geometry_msgs/Pose:
 geometry_msgs/Point position
 geometry_msgs/Quaternion orientation
```

La posición del robot se representa en el mensaje *Point* con sus coordenadas cartesianas. Al moverse sobre un plano las que se utilizan son x e y.

```
geometry_msgs/Point:
 float64 x
 float64 y
 float64 z
```

El mensaje *Quaternion* representa la orientación del robot utilizando un cuaternión, no los ángulos de Euler. Una vez se extraen éstos, el único que se utiliza es el correspondiente a la dirección normal al movimiento.

```
geometry_msgs/Quaternion:
 float64 x
 float64 y
 float64 z
 float64 w
```

Los elementos que se usan de la matriz de covarianzas de la posición son:

$$\Sigma = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & 0 & 0 & 0 & \sigma_{x\theta} \\ \sigma_{xy} & \sigma_y^2 & 0 & 0 & 0 & \sigma_{y\theta} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \sigma_{x\theta} & \sigma_{y\theta} & 0 & 0 & 0 & \sigma_\theta^2 \end{pmatrix},$$

y se almacenan en el vector *covariance* ordenados por filas.

La información sobre la velocidad del robot y su modelo gaussiano de error se incluye en *TwistWithCovariance*. Al igual que en el mensaje *PoseWithCovariance*, la matriz de covarianzas se reduce a 3×3.

```
geometry_msgs/TwistWithCovariance:
 geometry_msgs/Twist twist
 float64[36] covariance
```

Las velocidades lineal y angular se incluyen como vectores de tres elementos en el mensaje *Twist*. Para la velocidad lineal en el plano se utilizan los dos primeros elementos del vector *linear*, mientras que para la rotación sobre el eje perpendicular al plano se usa el último elemento de *angular*.

```
geometry_msgs/Twist:
 geometry_msgs/Vector3 linear
 geometry_msgs/Vector3 angular
```

El mensaje *Vector3* representa simplemente un vector de tres elementos.

```
geometry_msgs/Vector3:
 float64 x
 float64 y
 float64 z
```

ANEXO C

ANÁLISIS DE SENSIBILIDAD

En este anexo se describen las pruebas realizadas comprobar la sensibilidad del simulador ante variaciones de distintos parámetros. Estos parámetros son el radio de las ruedas, la masa y el tensor de inercia del robot, la posición de la masa del brazo robótico y el coeficiente de rozamiento de las ruedas con el suelo.

La tabla C.1 contiene todos los valores que toman los parámetros en las distintas pruebas.

Para evaluar los efectos de esas variaciones se medirá la distancia total recorrida por el robot, su velocidad media y su velocidad angular media en cuatro pruebas distintas. En dos las trayectorias serán rectas y en las otras dos serán ochos. Una de cada se realizará a una velocidad lenta y la otra rápida. El motivo para realizar un ocho es que se da una vuelta completa tanto hacia la derecha como hacia la izquierda.

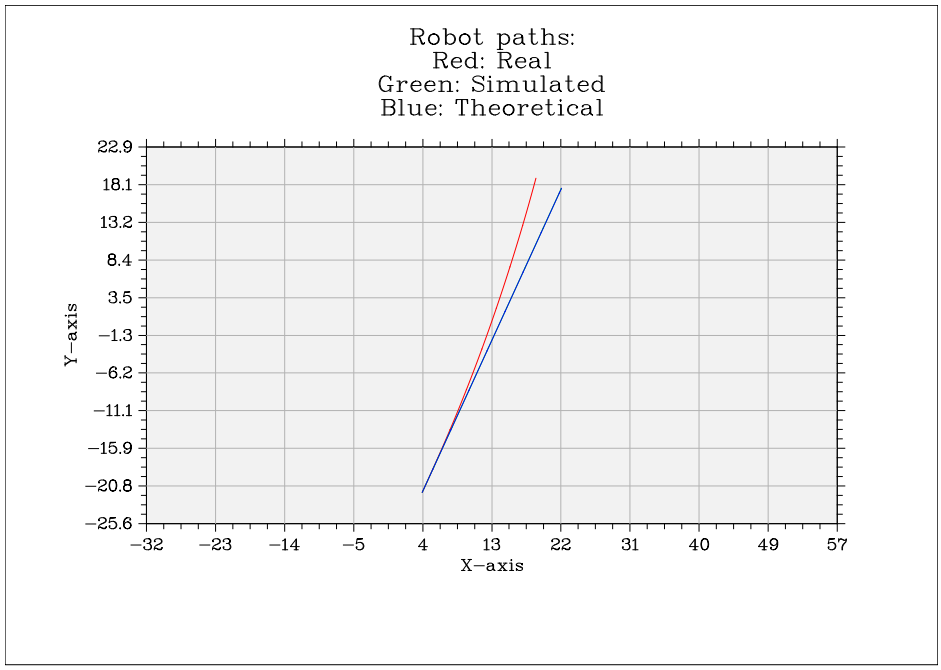
Las figuras C.1 y C.2 muestran las trayectorias realizadas para la primera de las pruebas. Todas ellas se han realizado a partir de órdenes de una prueba con el robot real. Además de la trayectoria simulada se muestra la que tendría que recorrer teóricamente a partir de esas órdenes y la que recorrió en realidad.

Hay que destacar que las velocidades angulares medias que se presentan se calculan utilizando valores absolutos. En caso contrario, los valores para un ocho serían cercanos a cero, al dar una vuelta en cada sentido.

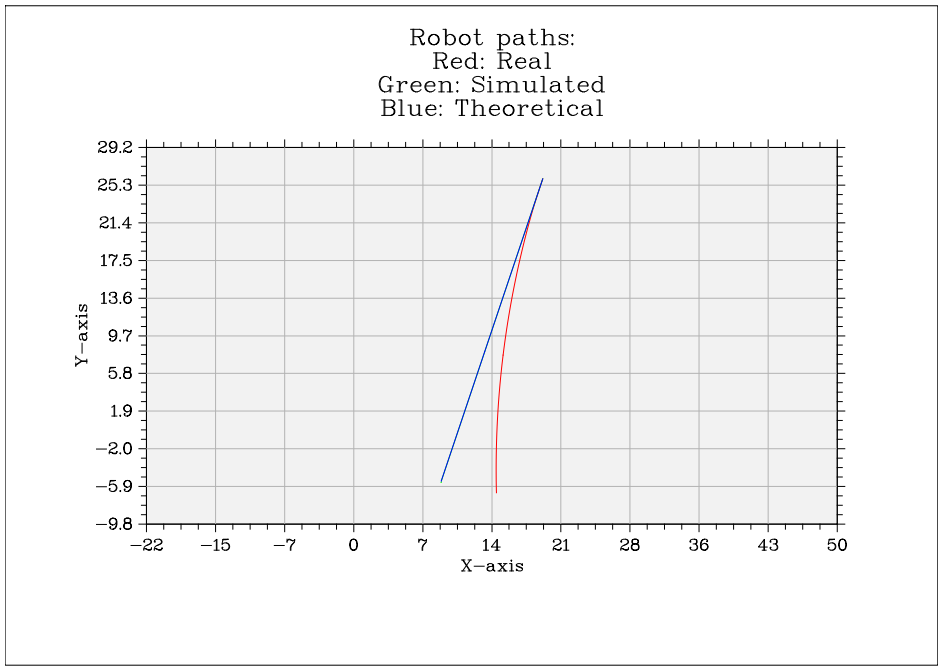
Las magnitudes medidas se redondean a la cuarta cifra significativa, que es una precisión para realizar este análisis.

C.1. Distribución de masa

Se realizan tres ensayos distintos relacionados con la masa del robot y su simulación.

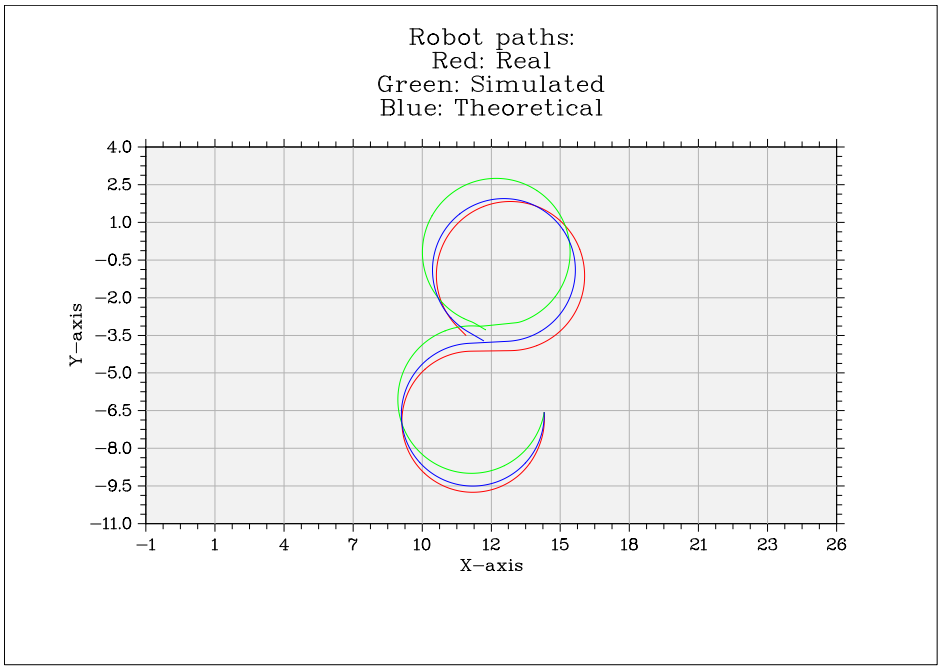


(a) Recta lenta

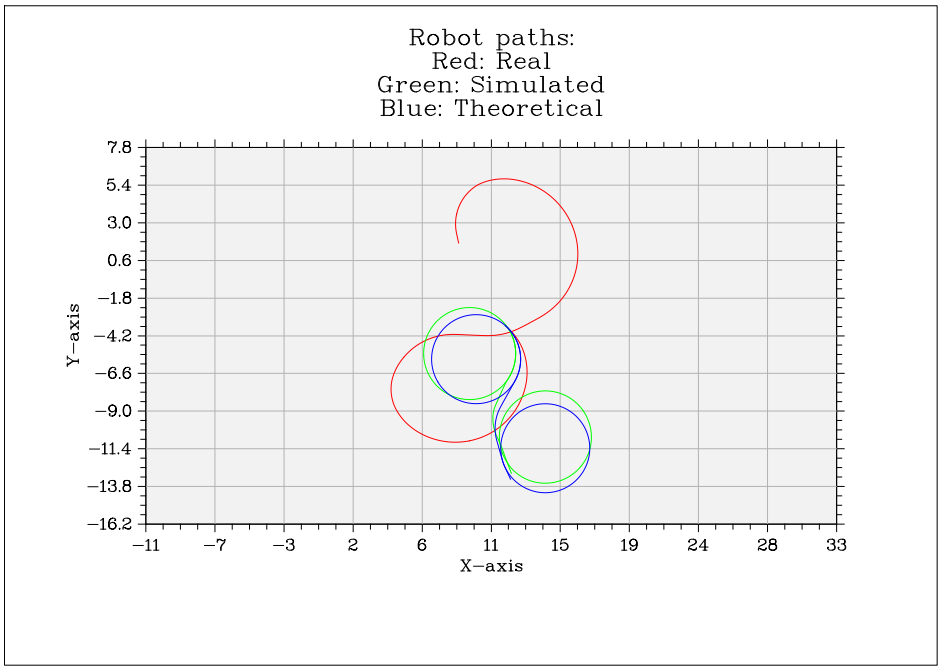


(b) Recta rápida

Figura C.1: Trayectorias rectas utilizadas para el análisis de sensibilidad de los parámetros del robot. En azul las órdenes enviadas, en verde la trayectoria simulada y en rojo la que realizó el robot real.



(a) Curva lenta



(b) Curva rápida

Figura C.2: Trayectorias curvas utilizadas para el análisis de sensibilidad de los parámetros del robot. En azul las órdenes enviadas, en verde la trayectoria simulada y en rojo la que realizó el robot real.

Parámetro	Valor elegido	Valores probados
Masa robot	200 kg	150; 250
Posición masa brazo	1	0,5; 0
Momentos de inercia:		
I_x	42,58 kg m^2	4,258; 425,8
I_y	154,58 kg m^2	15,458; 1545,8
I_z	168,33 kg m^2	16,833; 1683,3
Coeficiente de rozamiento	0,9	0,5; 100
Radios de las ruedas:		
Delantera izquierda	0,3 m	0,295
Delantera derecha	0,3 m	0,305
Trasera izquierda	0,3 m	0,295
Trasera derecha	0,3 m	0,305

Tabla C.1: Parámetros que serán modificados en el análisis de sensibilidad del simulador.

En el primero se cambia la masa total del robot, utilizando los valores entre lo que se cree que está. Los resultados se muestran en la tabla C.2. Se puede apreciar que en ninguno de los casos se produce ninguna variación importante. Es cierto que las velocidades angulares de las rectas cambian bastante, pero esto se debe únicamente a que son valores muy cercanos a cero, a efectos prácticos pueden ser considerados nulos.

En la segunda prueba lo que se cambia es la posición de la masa que representa la posición del brazo robótico. La posición 1 representa que está en la esquina superior izquierda, la 0 que está en el centro del robot y la 0,5 que está a medio camino entre las dos. Los resultados se presentan en la tabla C.3. Al igual que en caso anterior, no se descubren variaciones significativas.

En tercer lugar, la última prueba relacionada con la distribución de masa cambia el tensor de inercia. Los números indican el tensor que se utiliza de la tabla C.1. El 1 es para el original, el 2 es ése dividido entre diez y el 3 multiplicado por diez. Incluso utilizando unos valores extremos, no se produce ninguna variación importante.

C.2. Rozamiento

Se prueban coeficientes de rozamiento razonables para el contacto entre la goma de los neumáticos y distintas superficies, según [10]. Sin embargo, no hay

Trayectoria	Recta lenta			Recta rápida		
Masa	250	200	300	250	200	300
Distancia recorrida (m)	43,04	43,03	43,09	32,99	32,99	33,00
Velocidad media (m/s)	0,5136	0,5135	0,5148	1,641	1,641	1,642
Vel. angular media (rad/s)	0,0005671	0,0005235	0,0005495	0,0009577	0,001201	0,0008737
Trayectoria	Ocho lento			Ocho rápido		
Masa	250	200	300	250	200	300
Distancia recorrida (m)	32,35	32,34	32,36	47,05	47,04	47,16
Velocidad media (m/s)	0,4218	0,4222	0,4219	1,010	1,009	1,012
Vel. angular media (rad/s)	0,1326	0,1332	0,1327	0,3090	0,3070	0,3084

Tabla C.2: Resultados de las distintas pruebas cambiando la masa del robot.

Trayectoria	Recta lenta			Recta rápida		
Posición	1	0,5	0	1	0,5	0
Distancia recorrida (m)	43,04	43,07	43,09	32,99	32,99	32,99
Velocidad media (m/s)	0,5136	0,5140	0,5148	1,641	1,641	1,641
Vel. angular media (rad/s)	0,0005671	0,0003005	0,00009154	0,0009577	0,0004016	0,0001269
Trayectoria	Ocho lento			Ocho rápido		
Posición	1	0,5	0	1	0,5	0
Distancia recorrida (m)	32,35	32,30	32,39	47,05	47,13	47,11
Velocidad media (m/s)	0,4218	0,4211	0,4224	1,010	1,011	1,011
Vel. angular media (rad/s)	0,1326	0,1327	0,1330	0,3090	0,3090	0,3095

Tabla C.3: Resultados de las distintas pruebas cambiando la posición de la masa que simula el brazo robótico.

Trayectoria	Recta lenta			Recta rápida		
Tensor	1	2	3	1	2	3
Distancia recorrida (m)	43,04	43,05	43,07	32,99	32,99	33,00
Velocidad media (m/s)	0,5136	0,5137	0,5140	1,641	1,641	1,642
Vel. angular media (rad/s)	0,0005671	0,0006404	0,0001330	0,0009577	0,0009448	0,0003670
Trayectoria	Ocho lento			Ocho rápido		
Tensor	1	2	3	1	2	3
Distancia recorrida (m)	32,35	32,30	32,30	47,05	47,28	47,07
Velocidad media (m/s)	0,4218	0,4211	0,4217	1,010	1,017	1,010
Vel. angular media (rad/s)	0,1326	0,1326	0,1326	0,3090	0,3119	0,3088

Tabla C.4: Resultados de las distintas pruebas cambiando el tensor de inercia del robot.

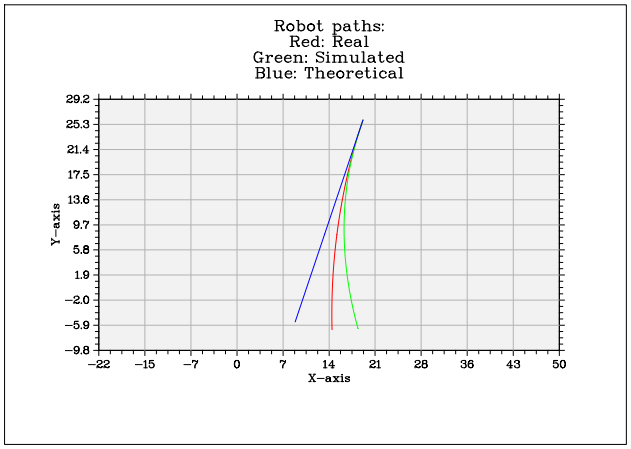


Figura C.3: Trayectorias efectuadas tanto por el robot simulado, en verde, como por el robot real, en rojo, en la prueba con variación de radios de las ruedas. La trayectoria teórica se muestra en azul.

que olvidar que hay mucho más parámetros que influyen en la fuerza de fricción. Utilizar un valor razonable no garantiza que los resultados sean realistas, así que se añade una prueba con un valor muy elevado. De esta forma, se puede comparar con un caso en el que no se produce deslizamiento.

Los resultados se muestran en la tabla C.5. Al igual que en las demás pruebas, no se aprecian cambios significativos. Hay que insistir que esto no implica que el rozamiento no sea importante en el robot real, únicamente que para imitar su comportamiento no se puede utilizar este parámetro, tal y como está diseñado el modelo.

C.3. Radios de las ruedas

La tabla C.6 presenta los resultados de la última de las pruebas. La prueba original se muestra bajo el número 1 y la otra bajo el 2. En esta segunda prueba se reducen ligeramente los radios de las ruedas del lado izquierdo y se aumentan las del derecho.

Se aprecia una importante variación en la velocidad angular de las trayectorias rectas, que provoca que la trayectoria del robot se aleje bastante de la recta que tendría que recorrer teóricamente, como se puede ver en la figura C.3. También se puede destacar que el robot real también presenta este comportamiento, aunque en menor medida.

Trayectoria		Recta lenta			Recta rápida	
Rozamiento	0,9	0,5	100	0,9	0,5	100
Distancia recorrida (m)	43,04	43,04	43,09	32,99	33,00	33,03
Velocidad media (m/s)	0,5136	0,5143	0,5141	1,641	1,642	1,643
Vel. angular media (rad/s)	0,0005671	0,0005224	0,0005692	0,0009577	0,0008564	0,001058
Trayectoria		Ocho lento			Ocho rápido	
Rozamiento	0,9	0,5	100	0,9	0,5	100
Distancia recorrida (m)	32,35	32,34	32,33	47,05	47,05	47,07
Velocidad media (m/s)	0,4218	0,4222	0,4220	1,010	1,010	1,012
Vel. angular media (rad/s)	0,1326	0,1331	0,1330	0,3090	0,3083	0,3080

Tabla C.5: Resultados de las distintas pruebas cambiando el rozamiento de las ruedas con el suelo.

Trayectoria	Recta lenta		Recta rápida	
Radios	1	2	1	2
Distancia recorrida (m)	43,04	43,02	32,99	32,97
Velocidad media (m/s)	0,5136	0,5134	1,641	1,632
Vel. angular media (rad/s)	0,0005671	0,006631	0,0009577	0,03070
Trayectoria	Ocho lento		Ocho rápido	
Radios	1	2	1	2
Distancia recorrida (m)	32,35	32,41	47,05	47,21
Velocidad media (m/s)	0,4218	0,4226	1,010	1,013
Vel. angular media (rad/s)	0,1326	0,1334	0,3090	0,3089

Tabla C.6: Resultados de las distintas pruebas cambiando los radios de las ruedas.

ANEXO D

ESTRUCTURA DE NODOS EN ROS

En este anexo se describen los nodos de ROS utilizados para las pruebas del capítulo 4 y las comunicaciones que se producen entre ellos mediante *topics*.

En primer lugar, se reproduce la información almacenada en una *rosbag* de una prueba real del robot. La figura D.1 muestra un esquema de los nodos en ese caso.

El nodo cuyo nombre empieza por */play* es el que está leyendo la *rosbag* y publicando sus *topics*. Los más importantes son */car/out*, de tipo *CarLikeInfo*, y */localization/out*, de tipo *PoseWithCovarianceStamped*. En el anexo B se describen los distintos tipos de mensajes de ROS utilizados.

extract_vel_cmd almacena esa información, que luego será reproducida de nuevo con el formato adecuado para la entrada del robot simulado. El simulador, en el nodo *gazebo*, ya está activo, aunque todavía no recibe información y por lo tanto el robot no realiza ninguna acción.

simulation_control es el nodo que se encarga de procesar todos los datos que se reciben, al final del proceso. Hasta entonces, va almacenando la información que necesita. Por último, *convert_ gps* [44] transforma la posición del GPS en latitud y longitud a coordenadas cartesianas. Sin embargo, se ha considerado que la posición que proporciona el IMU es más fiable para las pruebas realizadas en este proyecto, por lo que finalmente no ha sido utilizada.

Una vez finaliza la reproducción de la *rosbag*, el esquema cambia, como se refleja en la figura D.2. El nodo *extract_vel_cmd* transforma los datos recibidos al formato de las órdenes del robot, publicando en el *topic /robucar/car/in* mensajes del tipo *CarLikeCmd*. También publica la información de odometría en */real_robot/odometry*, con el tipo *Odometry*.

Ahora el robot simulado en el nodo *gazebo* ya está recibiendo órdenes y moviéndose. Publica en */robucar/car_like_info* mensajes del tipo *CarLikeInfo* y en */robucar/odometry* del tipo *Odometry*. También se publica en */robu-*

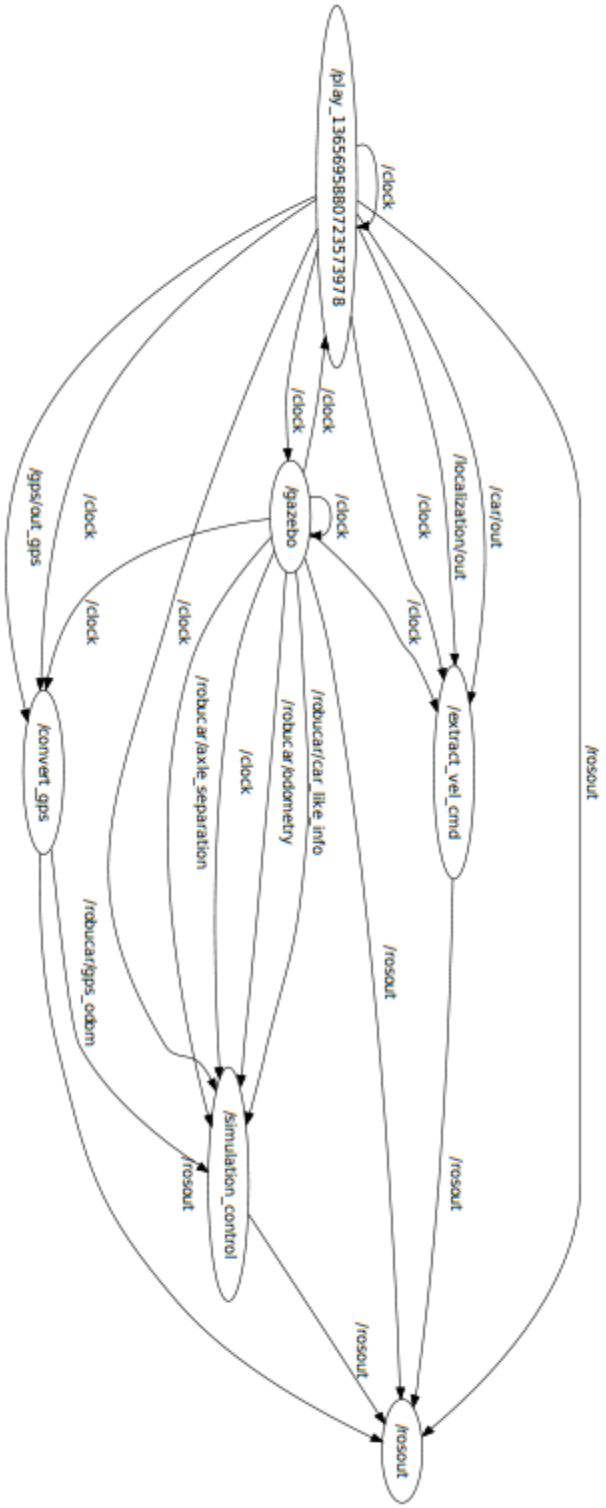


Figura D.1: Esquema de los nodos de ROS durante la reproducción de la información de una prueba.

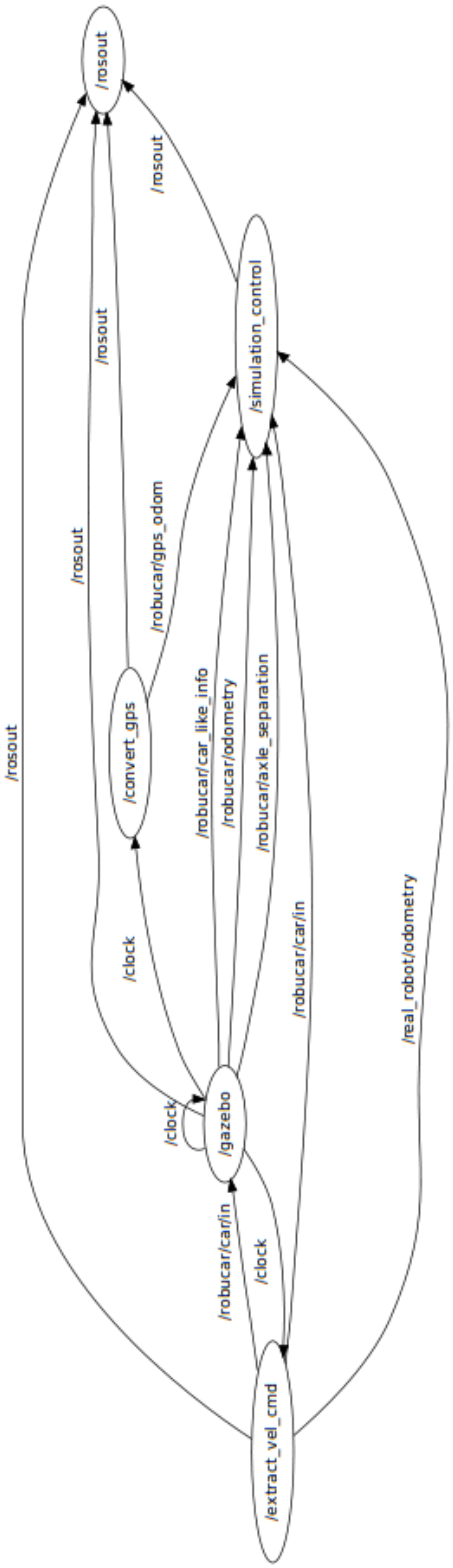


Figura D.2: Esquema de los nodos de ROS durante el procesado de la información de una prueba.

car/axle_separation la separación entre ejes del robot, necesaria para realizar algunos cálculos.

Hay dos razones para separar la lectura de la información de la *rosbag*. La primera es para poder transformar sin perder tiempo la información de salida del robot real en órdenes para el robot real. Al guardar la información directamente con el formato adecuado y posteriormente reproducirla en el instante de tiempo adecuado, no se producen más retrasos entre el momento en el que el robot real la publica y el simulado la recibe.

La otra razón es que se han encontrado problemas con *clock*, el *topic* que informa sobre el tiempo en la simulación, al existir varios nodos que publican en él.

ANEXO E

PRUEBAS DEL ROBOT

En este anexo se presentan los resultados de varias pruebas realizadas con el robot real, comparando su comportamiento con el que tiene la simulación recibiendo las mismas órdenes.

En primer lugar, se empezó utilizando Matlab para procesar la información, por medio de un *IPC bridge* [16], que permite la comunicación entre nodos de ROS y Matlab. Sin embargo, es un método lento y poco estable, por lo que se desechó y se pasó a escribir dos nodos de ROS. Hay que tener en cuenta que para cada pequeña variación del simulador es necesario volver a ejecutar la prueba entera, así que el que sea veloz y robusto es muy importante.

Uno de los nodos de ROS recibe la información de la *rosbag* donde se almacenan los datos de la prueba real. Los mensajes de salida utilizados son *CarLikeInfo* y *PoseWithCovariance*. De *CarLikeInfo* se pueden extraer las órdenes y crear un mensaje *CarLikeCmd*. También contiene información sobre la velocidad, que se puede combinar con la posición de *PoseWithCovariance* en un mensaje *Odometry*. Para ello se emparejan buscando los más cercanos en el tiempo, teniendo en cuenta que todos llevan información sobre cuándo se han generado. El anexo B explica en qué consisten estos mensajes.

Ese mensaje *Odometry* es recogido por el otro nodo, que registra las órdenes y la información de odometría del robot simulado. Con toda esta información puede dibujar las trayectorias de los robots real y simulado, además de calcular la trayectoria teórica que tendría que recorrer a partir de esas órdenes. También se registra información del GPS del robot, pero no se utiliza al haberse observado que, para las pruebas realizadas, la IMU proporciona resultados mucho mejores. En pruebas más largas el GPS sería más importante, ya que no acumula error.

Para buscar correlaciones entre variables se utilizan los coeficientes de Pearson y de Spearman. El primero indica correlación lineal, mientras que el segundo indica correlación en el orden de las variables [17]. Es importante no limitarse

a utilizar el coeficiente de Pearson, ya que el de Spearman permite encontrar las correlaciones no lineales que puedan existir. Sobre la significatividad de estas correlaciones, se realizan sobre una población bastante grande, con una muestra por cada décima de segundo.

Se realizan seis pruebas distintas para poder ajustar el robot. Dos de ellas son rectas efectuadas a velocidades distintas. Otras dos son simplemente giros con velocidades crecientes, cada uno en un sentido. Por último, se efectúan dos ochos, a velocidades distintas. Todas las pruebas se realizan sobre un suelo de grava. Además, se repiten tres de esas pruebas para comprobar el resultado del ajuste. Estas nueve pruebas se agrupan en rectas, giros y ochos.

Por último, se realizan dos pruebas más para observar cómo funciona el simulador en una condiciones para las que no ha sido ajustado específicamente.

En todas las gráficas presentadas el color rojo significa que es el robot real y el verde el simulado. El azul es el robot teórico computado a partir de las órdenes enviadas a los otros.

E.1. Rectas

La prueba 1 es una recta lenta y la prueba 2 una rápida.

En ambas trayectorias se puede observar que se produce una notable desviación hacia la izquierda, debido al menor radio de las ruedas de ese lado. Esto se refleja en la velocidad angular. En la recta lenta se observa con menos claridad debido a que las pequeñas variaciones lo ocultan, pero la figura E.7, correspondiente a la rápida, lo muestra con claridad. Por supuesto, en ambos casos la media del error de velocidad angular corrobora ese desvío, aún cuando en el primero se produce una desviación imporante.

Las velocidades lineales se ajustan casi a la perfección, como se ve tanto en las gráficas como en la media y dispersión del error. En el primer caso, tras la variación brusca de velocidad que se produce al principio, se observa un pico en la velocidad del robot simulado. Sin embargo, es muy pequeño y su duración es muy reducida, por lo que su efecto es mínimo.

Se observa, especialmente en el segundo caso, una correlación positiva relativamente fuerte entre la velocidad lineal del robot simulado y el error de velocidad angular. Parece que la importancia de esta desviación aumenta con la velocidad.

Para observar la corrección de la desviación hacia la izquierda, se repite la recta lenta tras el ajuste, llamada prueba 3.

Se observa que ahora existe una pequeña velocidad angular en la simulación, imitando la realidad gracias a los nuevos valores para los radios de las ruedas. La separación entre las posiciones finales se ha reducido considerablemente, y no hay ninguna correlación especialmente fuerte entre velocidades y errores. También existe un error en el ángulo, pero es prácticamente cero.

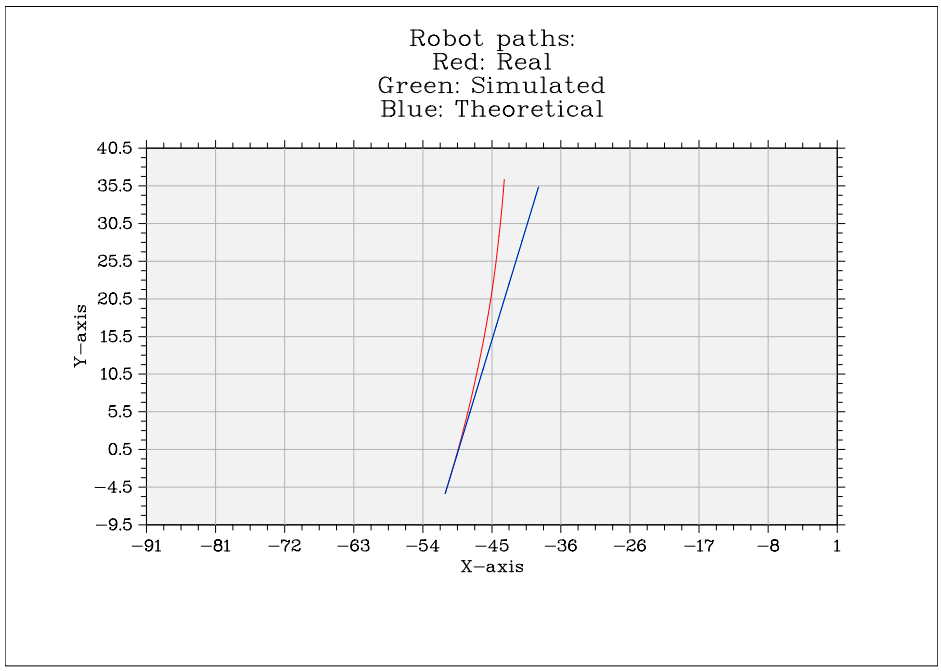


Figura E.1: Diferentes trayectorias de la prueba 1, recta lenta antes del ajuste.

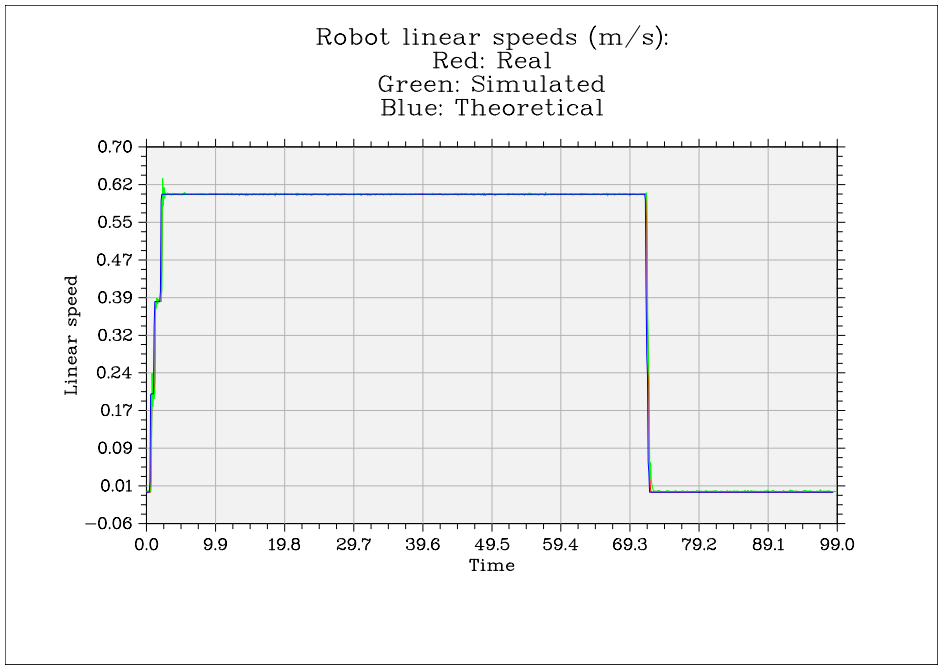


Figura E.2: Diferentes velocidades lineales de la prueba 1, recta lenta antes del ajuste.

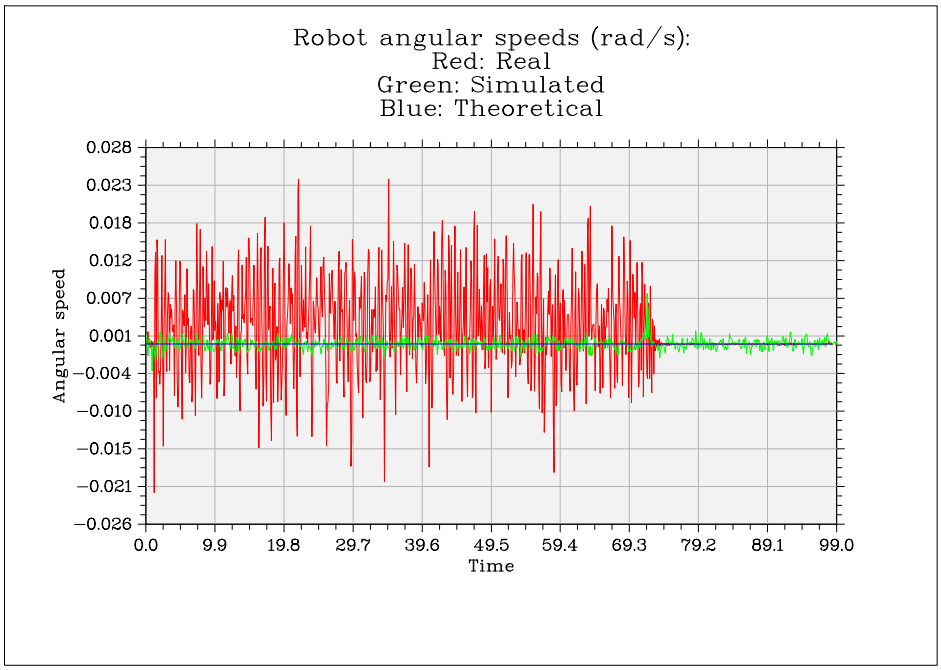


Figura E.3: Diferentes velocidades angulares de la prueba 1, recta lenta antes del ajuste.

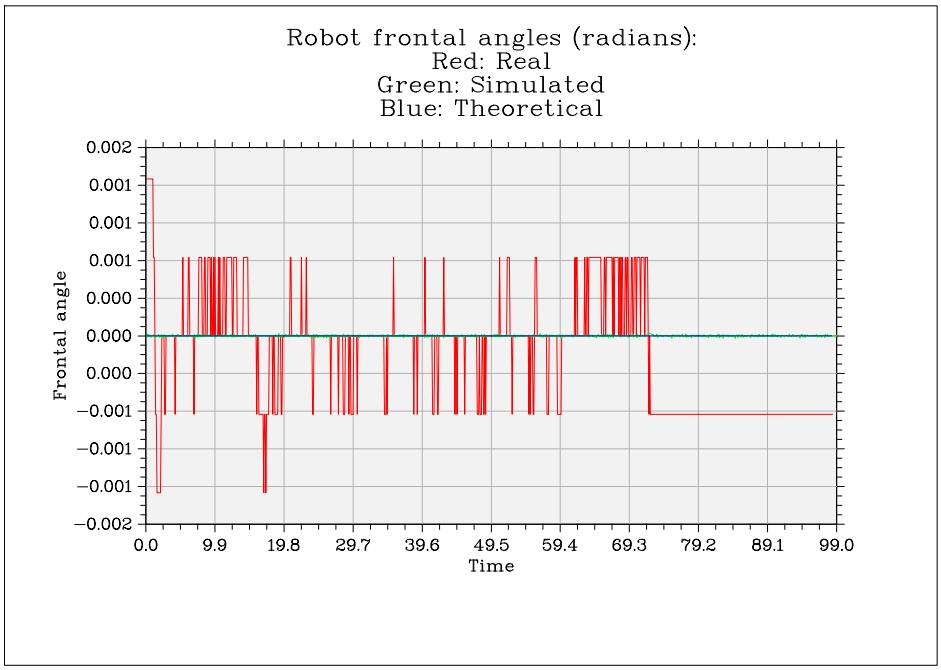


Figura E.4: Diferentes ángulos de la rueda virtual de la prueba 1, recta lenta antes del ajuste.

Robot	Real	Simulado	Teórico
Distancia recorrida (m)	42,49	42,53	42,49
Tiempo total (s)	98,4	98,4	98,3
Velocidad media (m/s)	0,4318	0,4322	0,4322
Vel. ang. media (rad/s)	0,004671	0,0005582	0
Comparación	Sim./Real	Sim./Teór.	Real/Teór.
Separación final (m)	4,637	0,02326	4,65
Separación por metro	0,109	0,000547	0,1094
Error vel. lineal (m/s)			
Media		-0,0006	
Desviación típica		0,0156	
Media abs.		0,0030	
Error vel. angular (rad/s)			
Media		0,0022	
Desviación típica		0,0064	
Media abs.		0,0048	
Error ángulo (rad)			
Media		0,0015	
Desviación típica		0,0000	
Media abs.		0,0015	
Correlación entre error vel. lineal y vel. lineal			
Pearson	-0,0760	Valor p	0,02361
Spearman	-0,2731	Valor p	1,533e-17
Correlación entre error vel. angular y vel. lineal			
Pearson	0,5365	Valor p	5,522e-73
Spearman	0,5367	Valor p	4,642e-73
Correlación entre error vel. lineal y vel. angular			
Pearson	0,1896	Valor p	7,062e-09
Spearman	0,0742	Valor p	0,0271
Correlación entre error vel. angular y vel. angular			
Pearson	0,0586	Valor p	0,07442
Spearman	0,1168	Valor p	0,000491

Tabla E.1: Resultados de la prueba 1, recta lenta antes del ajuste.

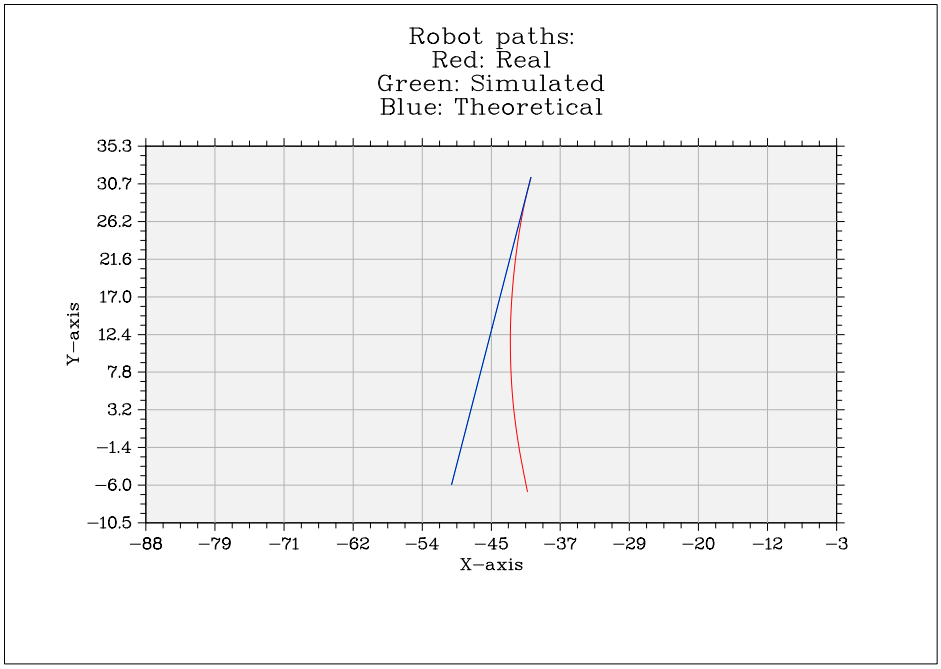


Figura E.5: Diferentes trayectorias de la prueba 2, recta rápida antes del ajuste.

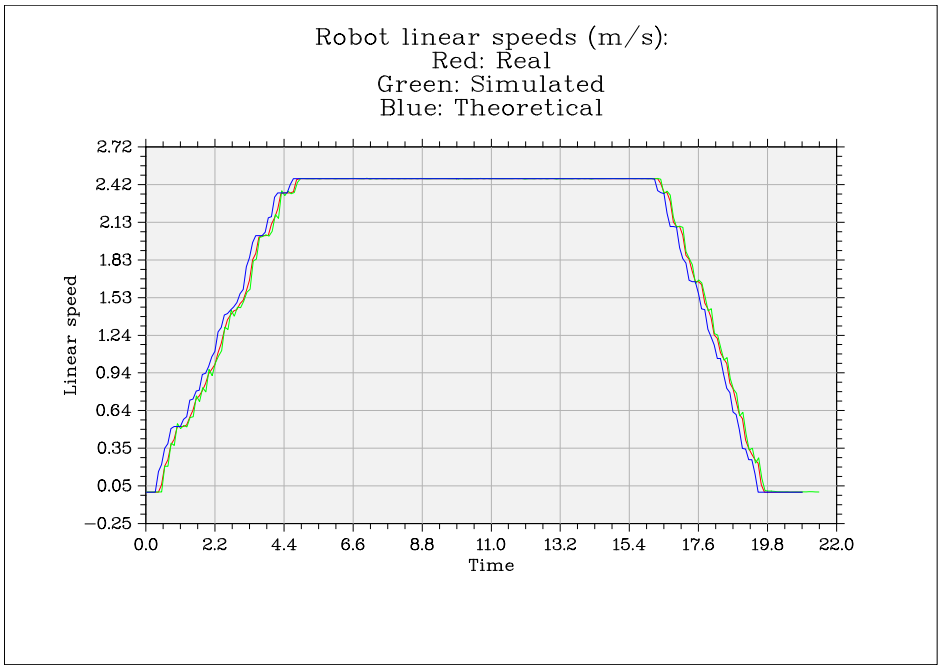


Figura E.6: Diferentes velocidades lineales de la prueba 2, recta rápida antes del ajuste.

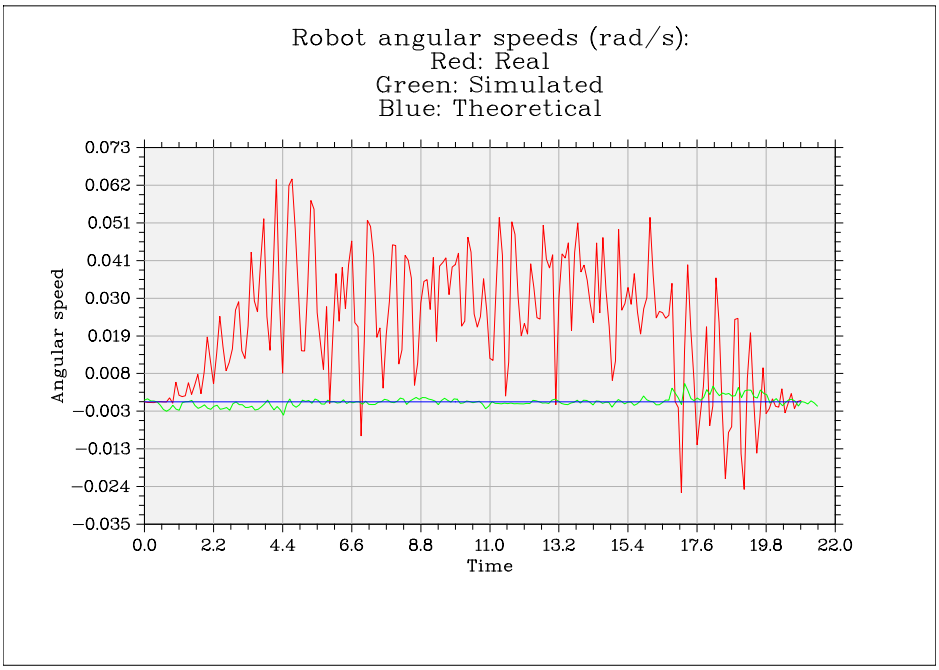


Figura E.7: Diferentes velocidades angulares de la prueba 2, recta rápida antes del ajuste.

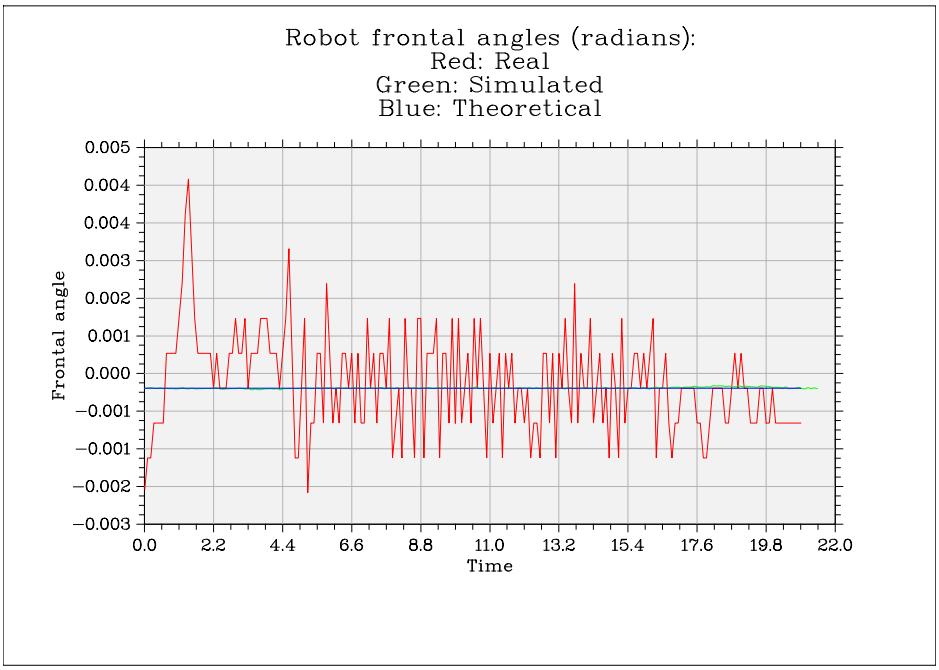


Figura E.8: Diferentes ángulos de la rueda virtual de la prueba 2, recta rápida antes del ajuste.

Robot	Real	Simulado	Teórico
Distancia recorrida (m)	38,6	38,59	38,6
Tiempo total (s)	20,9	20,9	20,9
Velocidad media (m/s)	1,847	1,847	1,847
Vel. ang. media (rad/s)	0,02344	0,0009438	0
Comparación	Sim./Real	Sim./Teór.	Real/Teór.
Separación final (m)	9,257	0,03026	9,273
Separación por metro	0,2399	0,000784	0,2403
Error vel. lineal (m/s)			
Media		0,0001	
Desviación típica		0,0645	
Media abs.		0,0317	
Error vel. angular (rad/s)			
Media		0,0217	
Desviación típica		0,0188	
Media abs.		0,0238	
Error ángulo (rad)			
Media		-0,0019	
Desviación típica		0,0004	
Media abs.		0,0019	
Correlación entre error vel. lineal y vel. lineal			
Pearson	-0,3413	Valor p	1,08e-06
Spearman	-0,5120	Valor p	8,598e-15
Correlación entre error vel. angular y vel. lineal			
Pearson	0,6942	Valor p	1,09e-30
Spearman	0,6102	Valor p	4,58e-22
Correlación entre error vel. lineal y vel. angular			
Pearson	0,3907	Valor p	1,429e-08
Spearman	0,5340	Valor p	3,226e-16
Correlación entre error vel. angular y vel. angular			
Pearson	-0,1805	Valor p	0,0129
Spearman	-0,1551	Valor p	0,03208

Tabla E.2: Resultados de la prueba 2, recta rápida antes del ajuste.

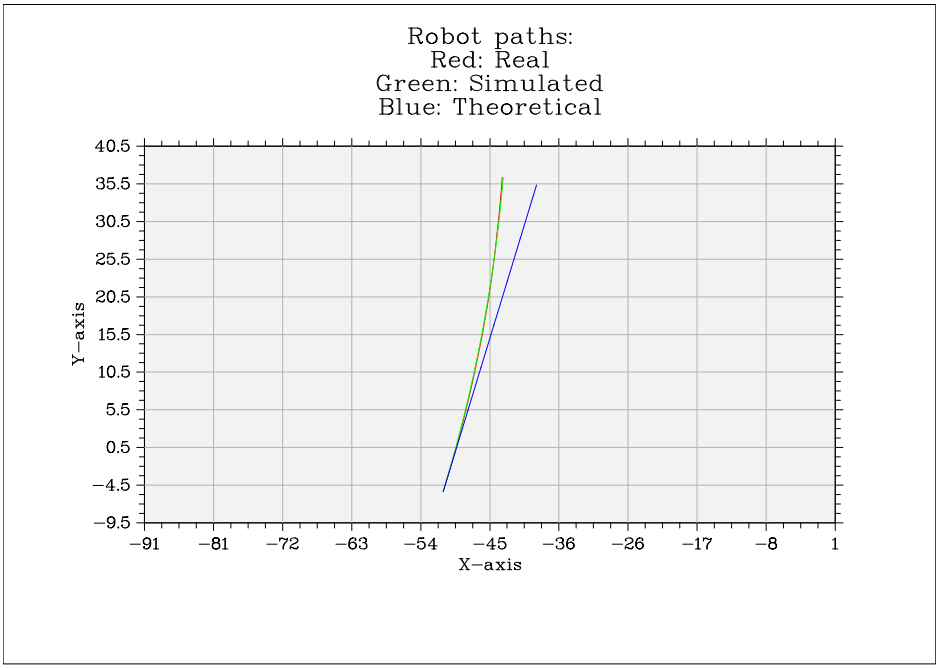


Figura E.9: Diferentes trayectorias de la prueba 3, recta lenta después del ajuste.

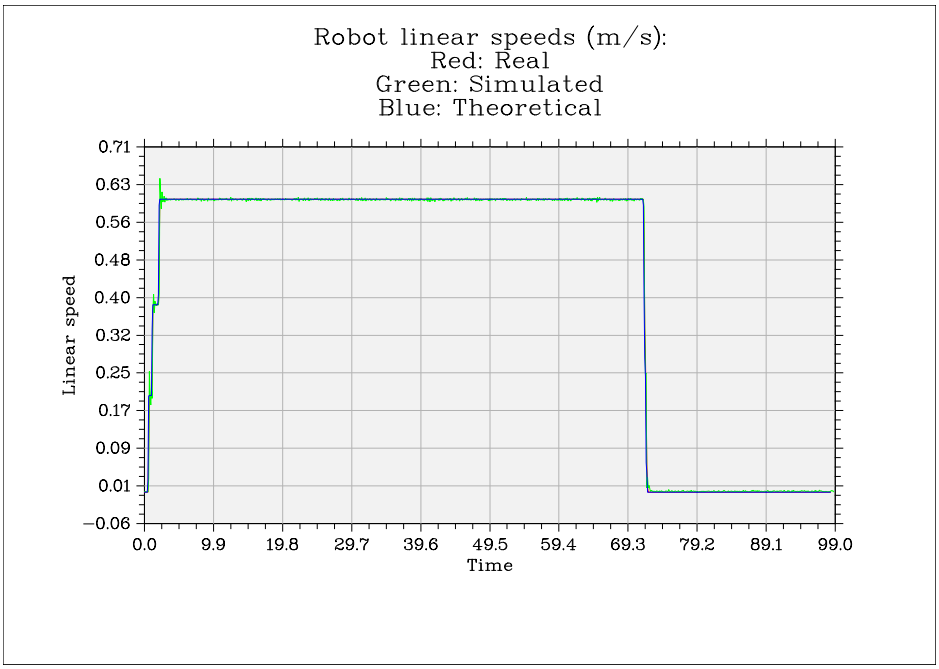


Figura E.10: Diferentes velocidades lineales de la prueba 3, recta lenta después del ajuste.

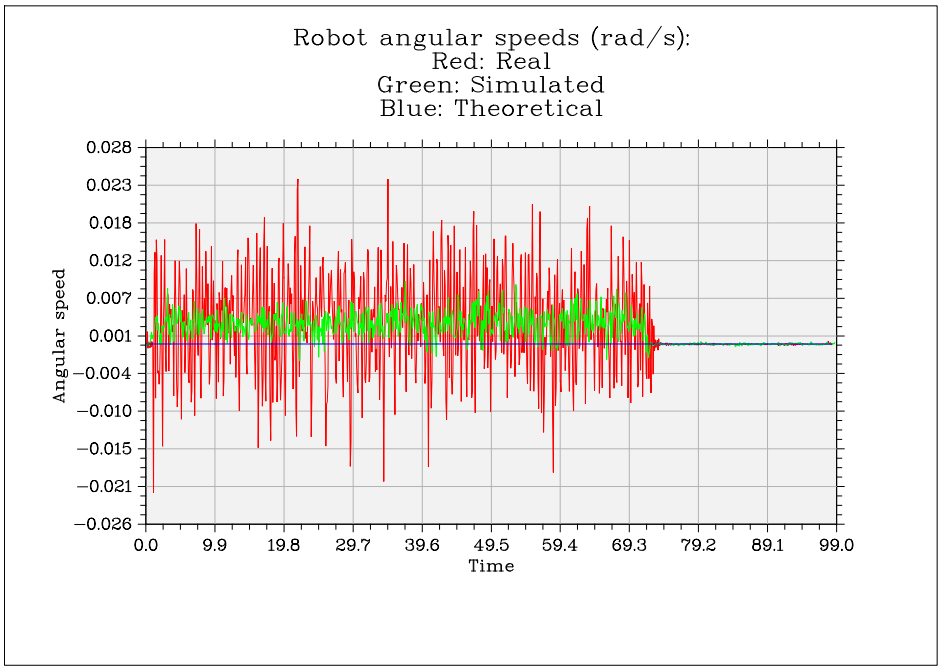


Figura E.11: Diferentes velocidades angulares de la prueba 3, recta lenta después del ajuste.

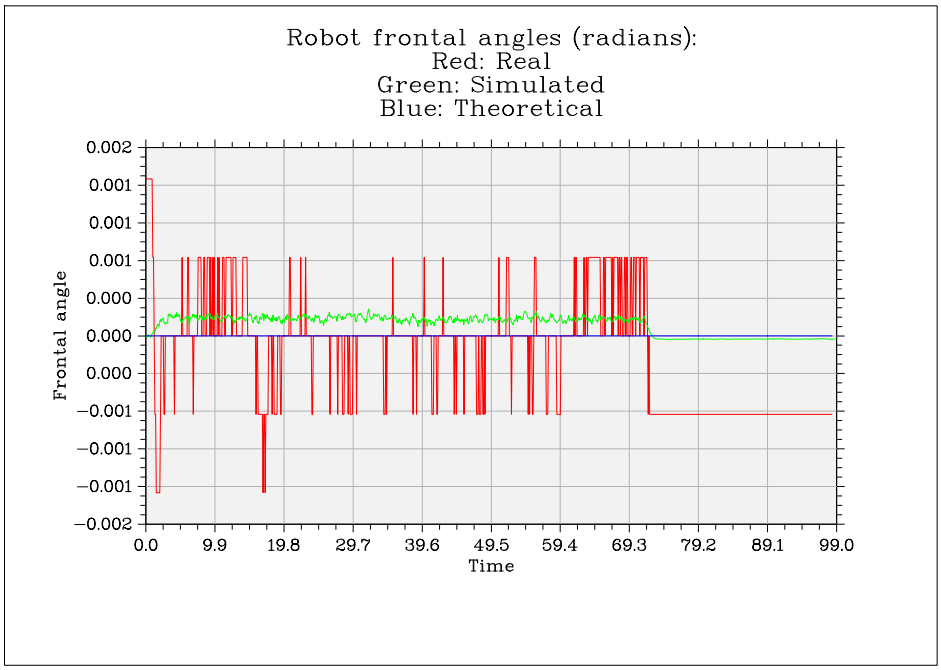


Figura E.12: Diferentes ángulos de la rueda virtual de la prueba 3, recta lenta después del ajuste.

Robot	Real	Simulado	Teórico
Distancia recorrida (m)	42,49	42,46	42,49
Tiempo total (s)	98,3	98,3	98,3
Velocidad media (m/s)	0,4322	0,432	0,4323
Vel. ang. media (rad/s)	0,004676	0,00225	0
Comparación	Sim./Real	Sim./Teór.	Real/Teór.
Separación final (m)	0,1277	4,525	4,65
Separación por metro	0,003007	0,1066	0,1095
Error vel. lineal (m/s)			
Media		0,0005	
Desviación típica		0,0117	
Media abs.		0,0027	
Error vel. angular (rad/s)			
Media		0,0001	
Desviación típica		0,0063	
Media abs.		0,0043	
Error ángulo (rad)			
Media		0,0014	
Desviación típica		0,0001	
Media abs.		0,0014	
Correlación entre error vel. lineal y vel. lineal			
Pearson	-0,0779	Valor p	0,02064
Spearman	-0,2791	Valor p	2,771e-18
Correlación entre error vel. angular y vel. lineal			
Pearson	0,5425	Valor p	7,286e-75
Spearman	0,5799	Valor p	7,806e-88
Correlación entre error vel. lineal y vel. angular			
Pearson	-0,0694	Valor p	0,03807
Spearman	-0,1258	Valor p	0,0001696
Correlación entre error vel. angular y vel. angular			
Pearson	0,3767	Valor p	1,735e-33
Spearman	0,5765	Valor p	1,443e-86

Tabla E.3: Resultados de la prueba 3, recta lenta después del ajuste.

E.2. Giros

La prueba 4 corresponde a un giro hacia la izquierda y la prueba 5 a uno hacia la derecha.

Se intenta observar la importancia del deslizamiento en giros en ambos sentidos. En las dos pruebas se gira el máximo las ruedas del robot y se va incrementando gradualmente su velocidad lineal. En esta situación, al aumentarla también lo hace linealmente la velocidad angular. En teoría, para un ángulo de ruedas constante el radio de giro no debería variar, pero las trayectorias indican claramente que ha ido aumentando.

Además, las gráficas de velocidad angular muestran que, a partir de un punto, casi no aumenta más, y se queda muy lejos del valor teórico. Se observa una correlación lineal positiva muy fuerte entre el error en velocidad angular y las velocidades lineal y angular, en ambos casos.

Una diferencia muy clara es el error en velocidad lineal que se produce al girar a la derecha, con una correlación lineal positiva casi perfecta con las velocidades. En el giro hacia la izquierda la correlación lineal es mucho más débil, aunque siga existiendo según el coeficiente de Spearman. Este efecto se debe a la asimetría de la distribución de masas en el modelo del simulador.

Para comprobar la corrección del ajuste no se utilizan las pruebas de giros, ya que se considera que los ochos son más completos, al incorporar en la misma prueba un giro en ambos sentidos.

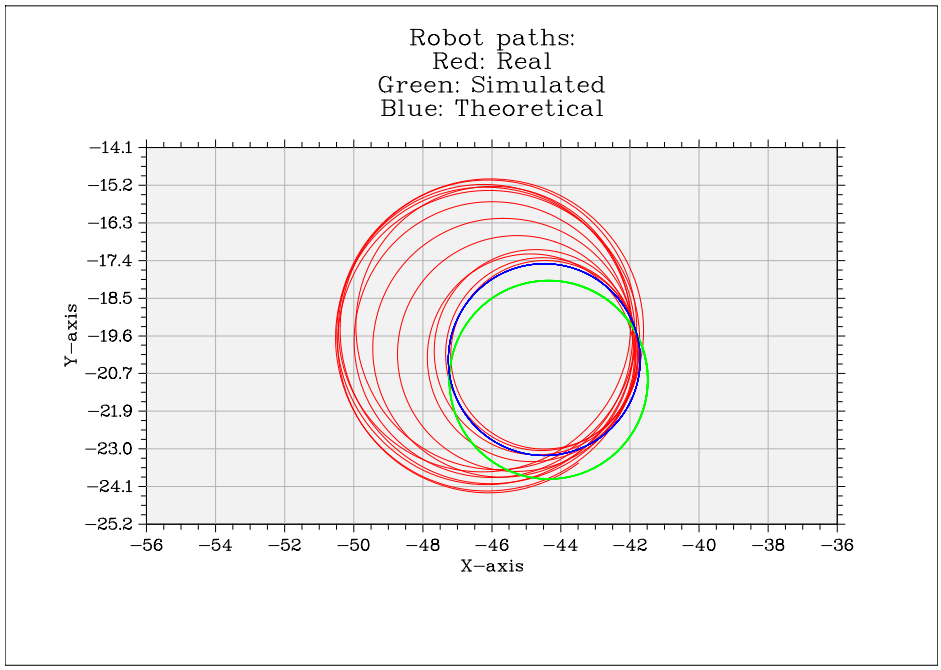


Figura E.13: Diferentes trayectorias de la prueba 4, giro hacia la izquierda antes del ajuste.

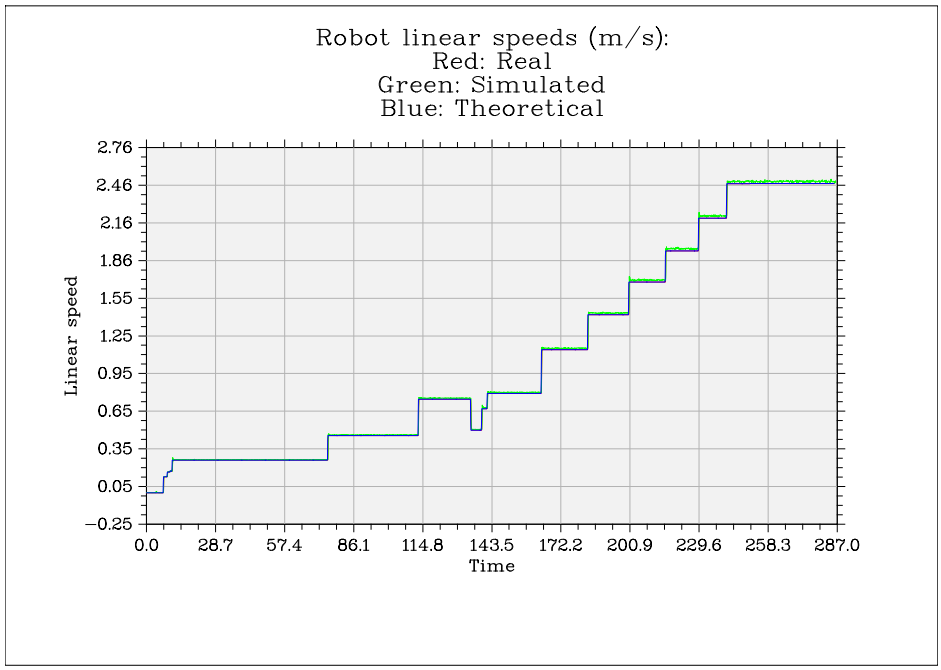


Figura E.14: Diferentes velocidades lineales de la prueba 4, giro hacia la izquierda antes del ajuste.

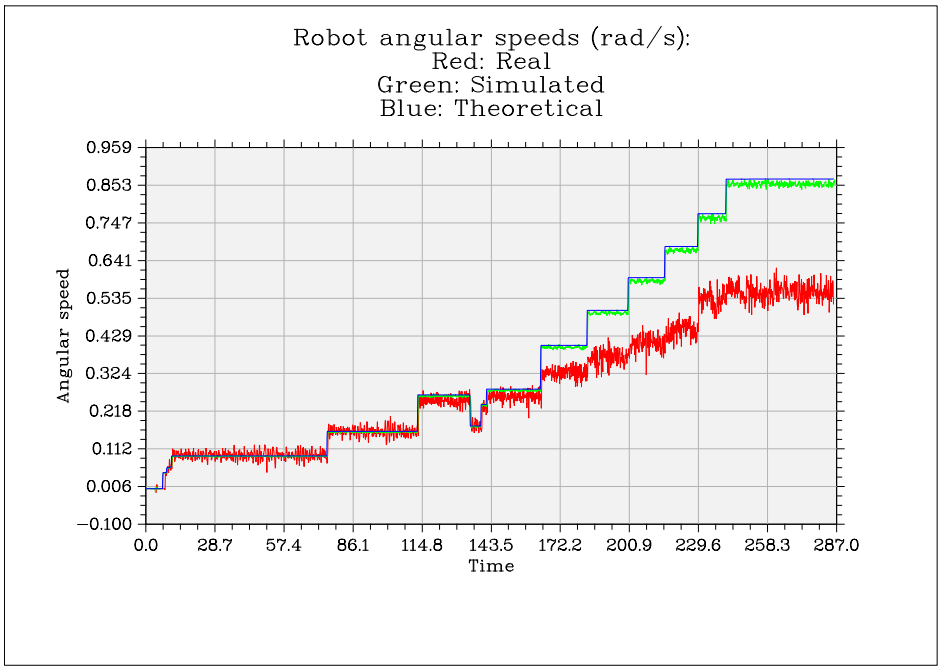


Figura E.15: Diferentes velocidades angulares de la prueba 4, giro hacia la izquierda antes del ajuste.

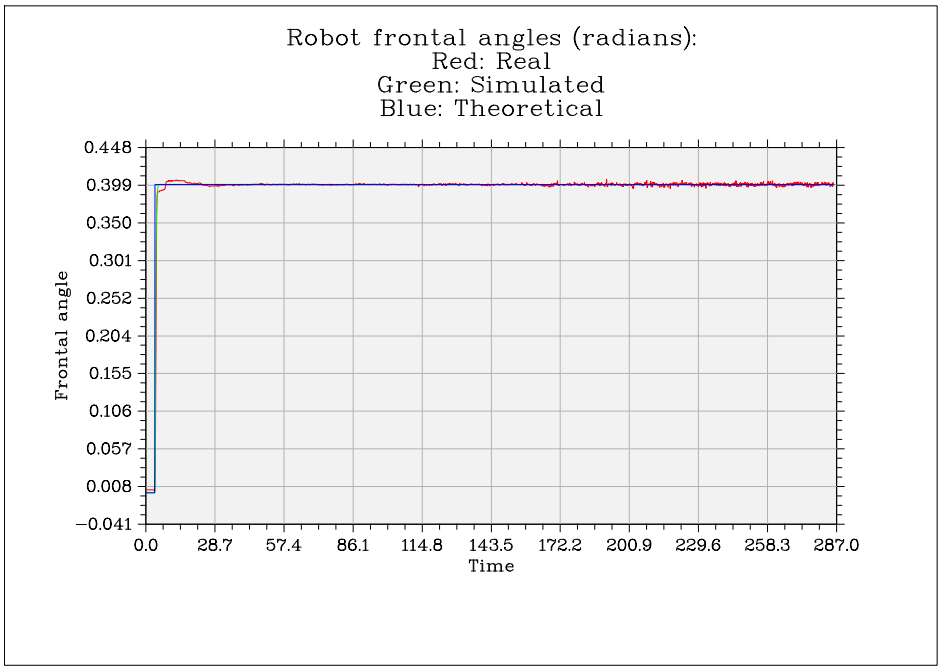


Figura E.16: Diferentes ángulos de la rueda virtual de la prueba 4, giro hacia la izquierda antes del ajuste.

Robot	Real	Simulado	Teórico
Distancia recorrida (m)	306,8	314,1	306,9
Tiempo total (s)	285,7	285,7	285,6
Velocidad media (m/s)	1,074	1,099	1,075
Vel. ang. media (rad/s)	0,2817	0,3736	0,3787
Comparación	Sim./Real	Sim./Teór.	Real/Teór.
Separación final (m)	4,512	3,035	6,009
Separación por metro	0,01436	0,009661	0,01959
Error vel. lineal (m/s)			
Media		-0,0091	
Desviación típica		0,0114	
Media abs.		0,0103	
Error vel. angular (rad/s)			
Media		-0,0907	
Desviación típica		0,1178	
Media abs.		0,0957	
Error ángulo (rad)			
Media		-0,3904	
Desviación típica		0,0474	
Media abs.		0,3905	
Correlación entre error vel. lineal y vel. lineal			
Pearson	0,5197	Valor p	1,537e-194
Spearman	0,9009	Valor p	0
Correlación entre error vel. angular y vel. lineal			
Pearson	0,9605	Valor p	0
Spearman	0,8611	Valor p	0
Correlación entre error vel. lineal y vel. angular			
Pearson	0,5179	Valor p	6,02e-193
Spearman	0,8529	Valor p	0
Correlación entre error vel. angular y vel. angular			
Pearson	0,9607	Valor p	0
Spearman	0,8605	Valor p	0

Tabla E.4: Resultados de la prueba 4, giro hacia la izquierda antes del ajuste.

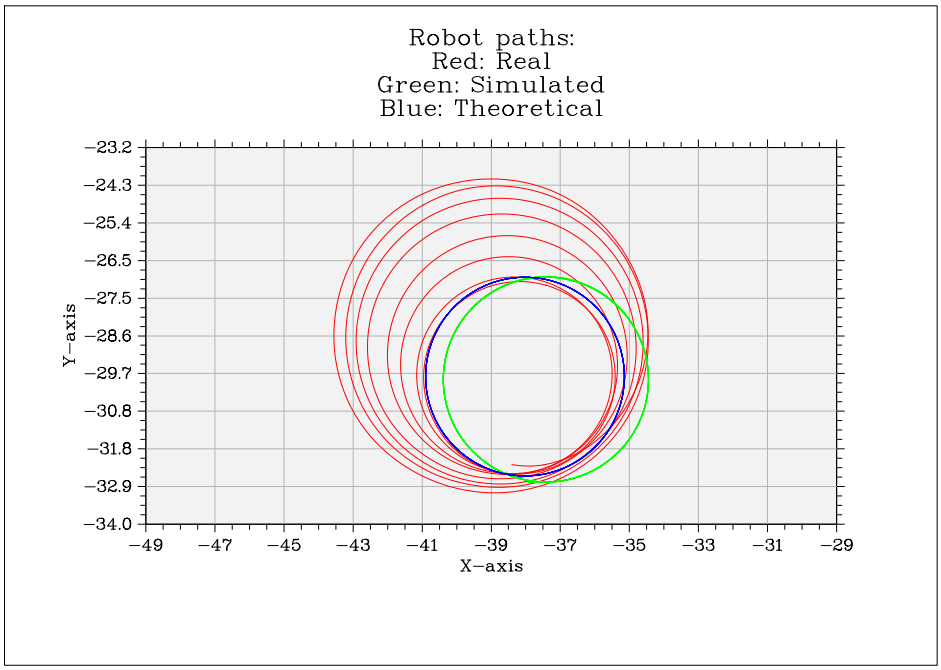


Figura E.17: Diferentes trayectorias de la prueba 5, giro hacia la derecha antes del ajuste.

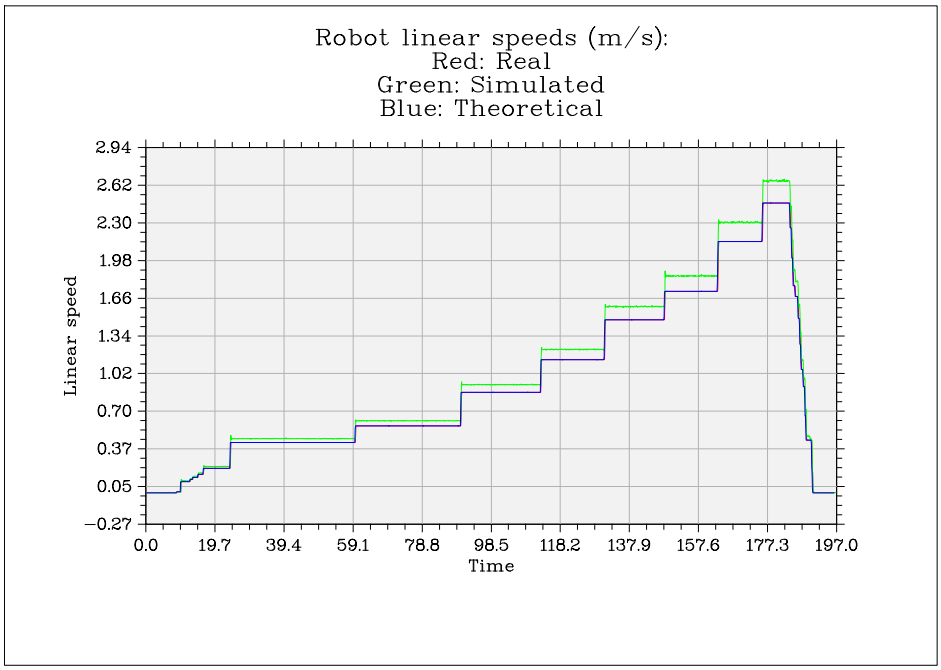


Figura E.18: Diferentes velocidades lineales de la prueba 5, giro hacia la derecha antes del ajuste.

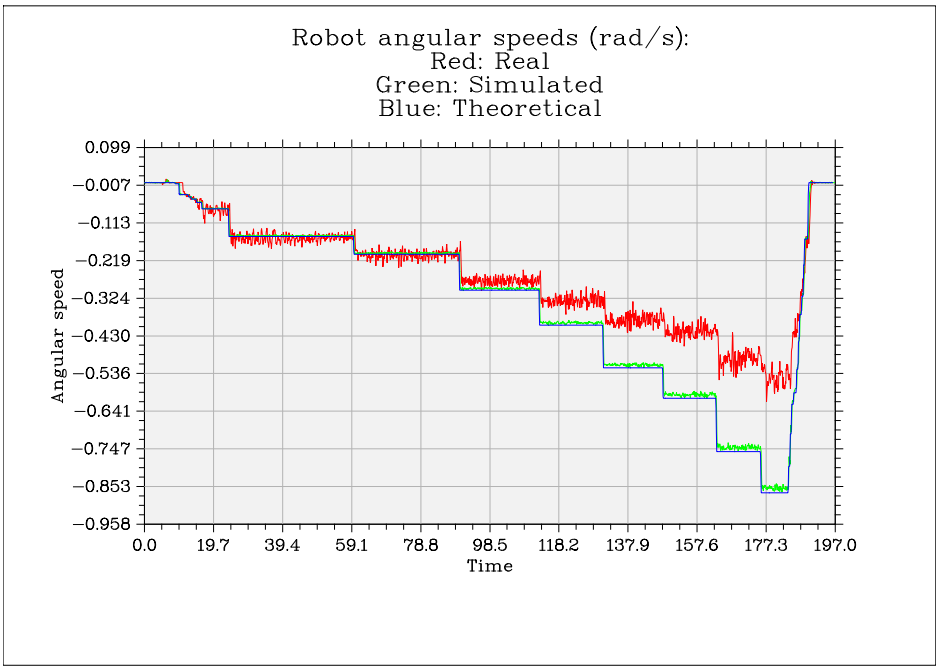


Figura E.19: Diferentes velocidades angulares de la prueba 5, giro hacia la derecha antes del ajuste.

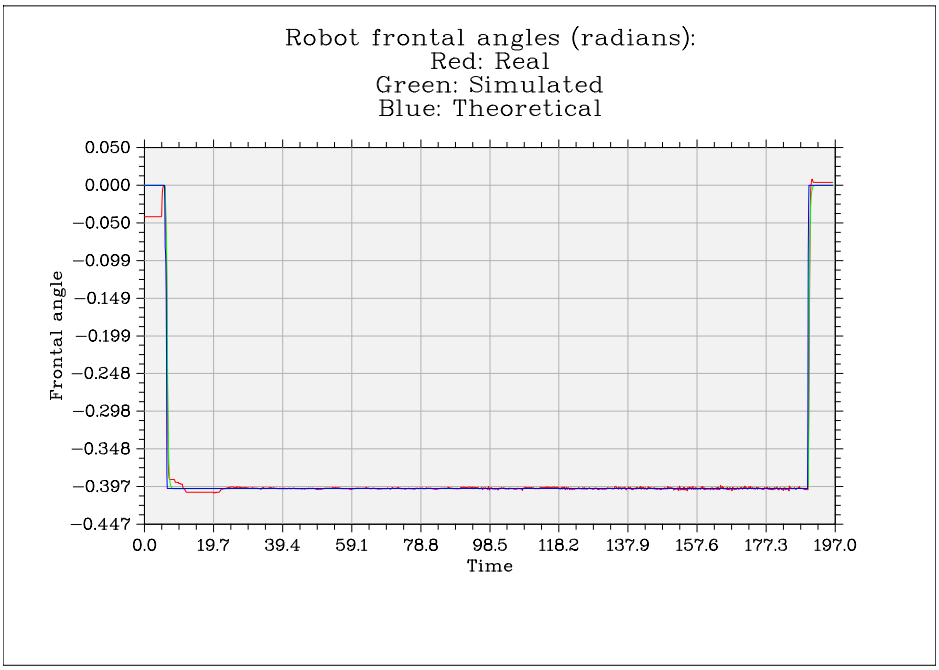


Figura E.20: Diferentes ángulos de la rueda virtual de la prueba 5, giro hacia la derecha antes del ajuste.

Robot	Real	Simulado	Teórico
Distancia recorrida (m)	180,5	183,5	180,5
Tiempo total (s)	196,2	196,2	196,2
Velocidad media (m/s)	0,9201	0,9351	0,92
Vel. ang. media (rad/s)	0,256	0,3178	0,3236
Comparación	Sim./Real	Sim./Teór.	Real/Teór.
Separación final (m)	0,6988	2,79	2,138
Separación por metro	0,003809	0,01521	0,01184
Error vel. lineal (m/s)			
Media		-0,0706	
Desviación típica		0,0525	
Media abs.		0,0708	
Error vel. angular (rad/s)			
Media		0,0626	
Desviación típica		0,0951	
Media abs.		0,0697	
Error ángulo (rad)			
Media		0,3306	
Desviación típica		0,1009	
Media abs.		0,3359	
Correlación entre error vel. lineal y vel. lineal			
Pearson	0,9946	Valor p	0
Spearman	0,9968	Valor p	0
Correlación entre error vel. angular y vel. lineal			
Pearson	0,9406	Valor p	0
Spearman	0,8834	Valor p	0
Correlación entre error vel. lineal y vel. angular			
Pearson	0,9942	Valor p	0
Spearman	0,9827	Valor p	0
Correlación entre error vel. angular y vel. angular			
Pearson	0,9393	Valor p	0
Spearman	0,8832	Valor p	0

Tabla E.5: Resultados de la prueba 5, giro hacia la derecha antes del ajuste.

E.3. Ochos

La prueba 6 corresponde a un ocho lento y la prueba 7 a uno más rápido.

Al comparar ambas pruebas se observa de nuevo el efecto del deslizamiento. En la rápida la trayectoria que realiza se aleja mucho de la que efectuaría un robot teórico, mientras que en la lenta son muy parecidas. La velocidad angular real de la prueba rápida, en la figura E.23, está muy por debajo de la que debería alcanzar. Además, existe una clara correlación entre el error de velocidad angular y las velocidades en ese caso, mientras que en el lento o no existe o es muy débil. No hay que olvidar tampoco que la media de ese error de velocidad angular es prácticamente cero en la prueba lenta y que toma un valor apreciable en la rápida.

También se observa de nuevo el error de velocidad lineal para giros a la derecha, en las dos pruebas.

Por último, se puede apreciar que el robot simulado tiende a girar más al principio del movimiento, alejando su trayectoria de la teórica.

Se repiten ambos ochos. Los nuevos ochos son las pruebas 8 y 9. Contienen giros en ambos sentidos a varias velocidades.

El ocho lento presenta un comportamiento bastante similar al anterior, ya que su velocidad angular es baja y no se ve afectado por la corrección del deslizamiento. Las trayectorias se separan al principio debido a una órden de ángulo a la que no responde el robot real y sí el simulado, pero después son parecidas. Se produce un pequeño error en la velocidad lineal simulada a causa de la variación de los radios, pero es pequeño.

En el ocho rápido se observan los mayores cambios. La trayectoria simulada se acerca más a la real, pero sigue siendo distinta, ya que las velocidades angulares reales son menores. El ajuste se ha realizado con los giros de la sección anterior, las pruebas 3 y 4, que se llevaron a cabo en un día distinto. Queda claro que el sistema de corrección de la velocidad angular tiene una efectividad muy limitada, y para ser utilizado tiene que ser reajustado para cada situación.

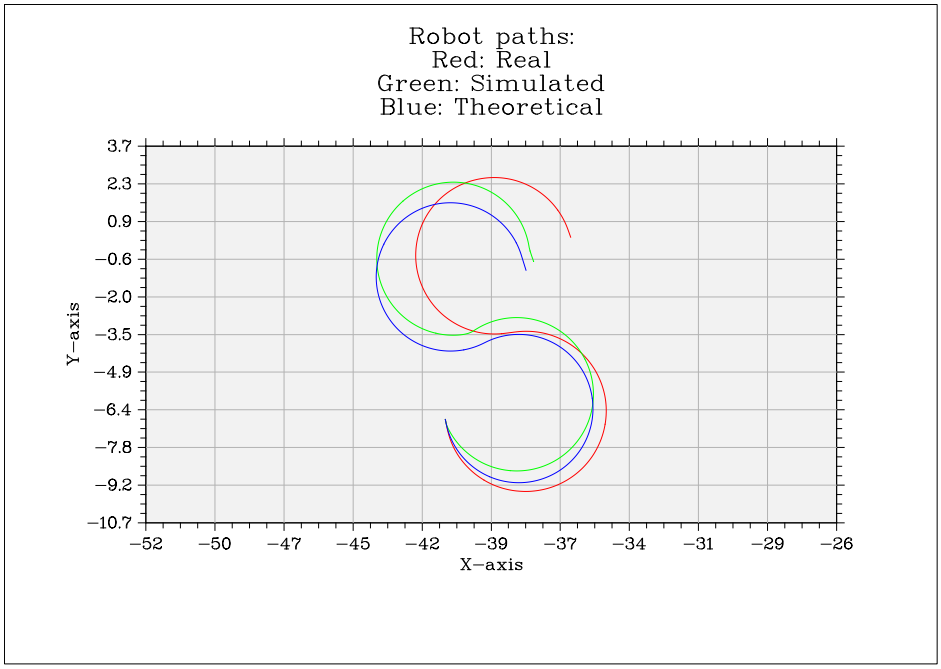


Figura E.21: Diferentes trayectorias de la prueba 6, ocho lento antes del ajuste.

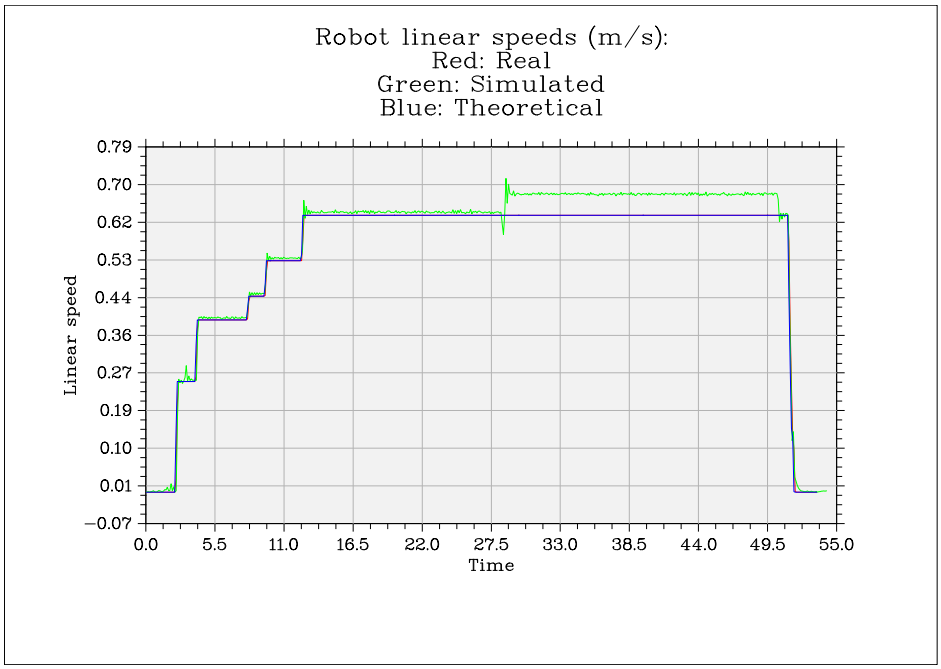


Figura E.22: Diferentes velocidades lineales de la prueba 6, ocho lento antes del ajuste.

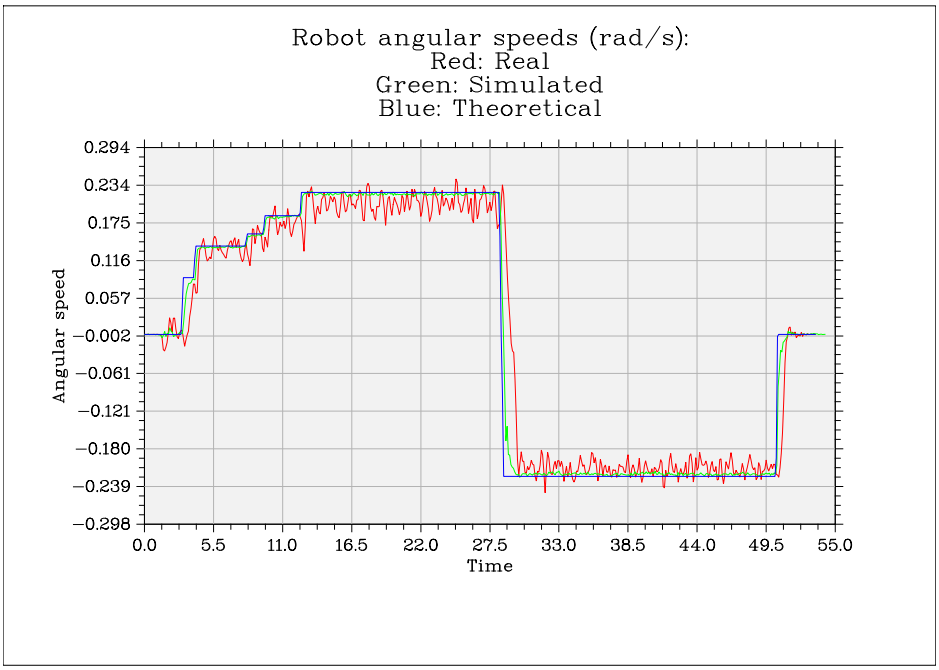


Figura E.23: Diferentes velocidades angulares de la prueba 6, ocho lento antes del ajuste.

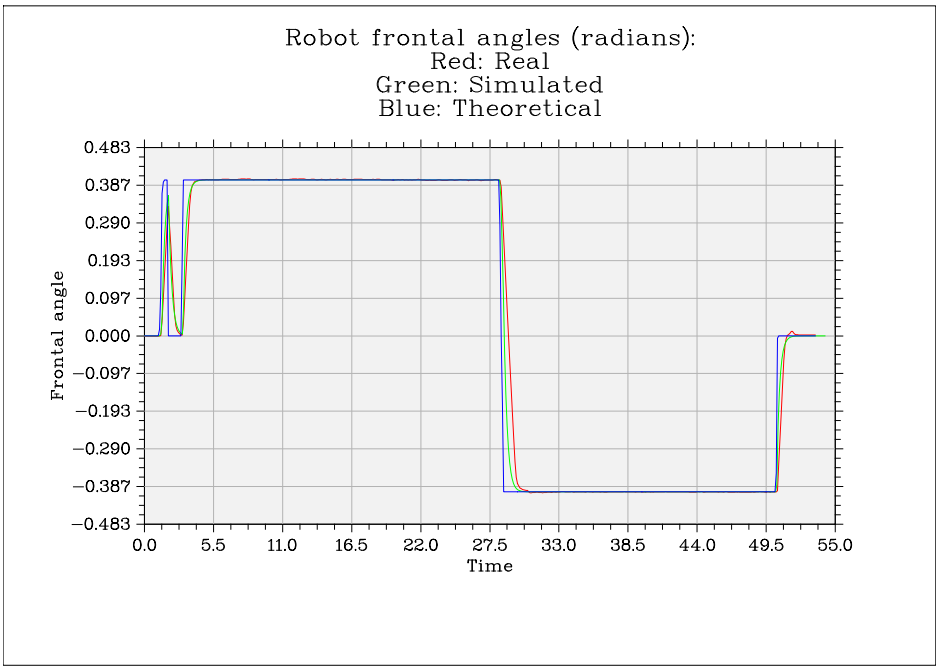


Figura E.24: Diferentes ángulos de la rueda virtual de la prueba 6, ocho lento antes del ajuste.

Robot	Real	Simulado	Teórico
Distancia recorrida (m)	28,81	29,34	28,81
Tiempo total (s)	53,4	53,4	53,41
Velocidad media (m/s)	0,5396	0,5494	0,5394
Vel. ang. media (rad/s)	0,1719	0,1785	0,1842
Comparación	Sim./Real	Sim./Teór.	Real/Teór.
Separación final (m)	1,695	0,4448	2,127
Separación por metro	0,05779	0,01516	0,07382
Error vel. lineal (m/s)			
Media		-0,0226	
Desviación típica		0,0251	
Media abs.		0,0249	
Error vel. angular (rad/s)			
Media		0,0002	
Desviación típica		0,0386	
Media abs.		0,0192	
Error ángulo (rad)			
Media		-0,0315	
Desviación típica		0,3703	
Media abs.		0,3497	
Correlación entre error vel. lineal y vel. lineal			
Pearson	0,4227	Valor p	8,898e-24
Spearman	0,9124	Valor p	9,292e-207
Correlación entre error vel. angular y vel. lineal			
Pearson	0,1545	Valor p	0,0006628
Spearman	0,1290	Valor p	0,004683
Correlación entre error vel. lineal y vel. angular			
Pearson	0,3256	Valor p	5,007e-14
Spearman	0,4043	Valor p	1,16e-21
Correlación entre error vel. angular y vel. angular			
Pearson	-0,0357	Valor p	0,2846
Spearman	0,2232	Valor p	5,227e-07

Tabla E.6: Resultados de la prueba 6, ocho lento antes del ajuste.

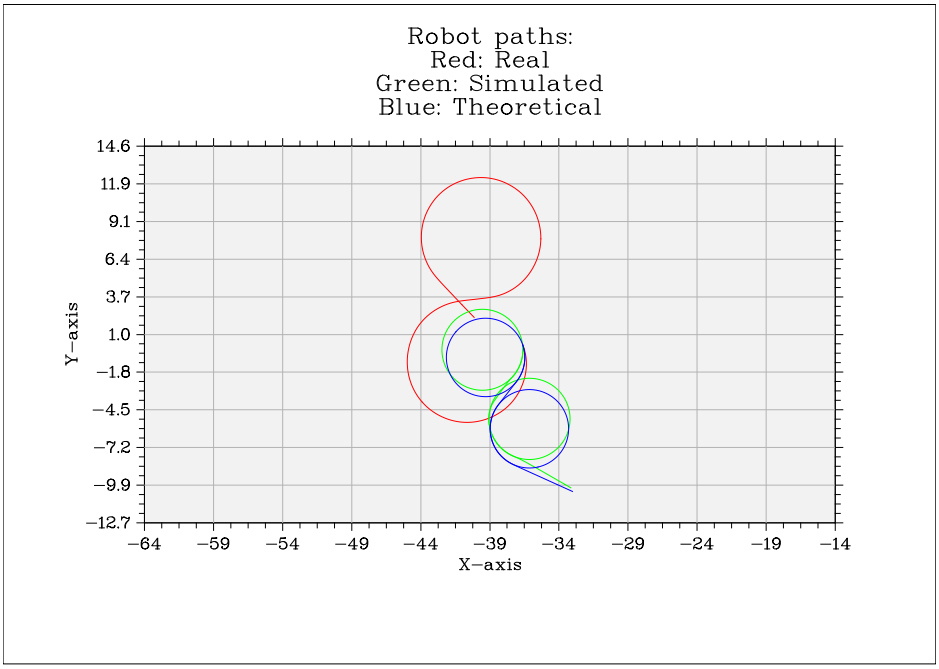


Figura E.25: Diferentes trayectorias de la prueba 7, ocho rápido antes del ajuste.

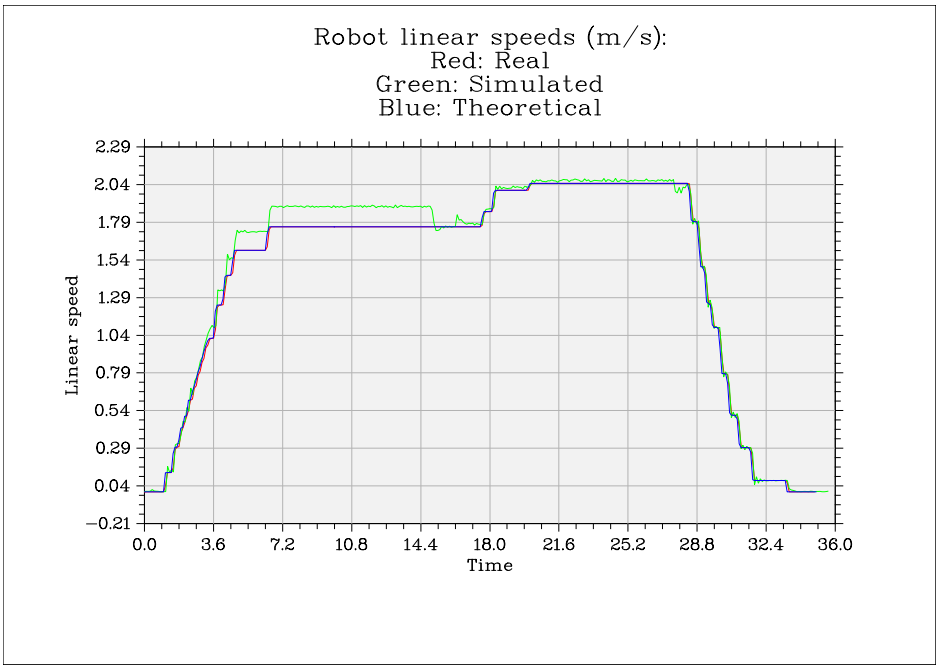


Figura E.26: Diferentes velocidades lineales de la prueba 7, ocho rápido antes del ajuste.

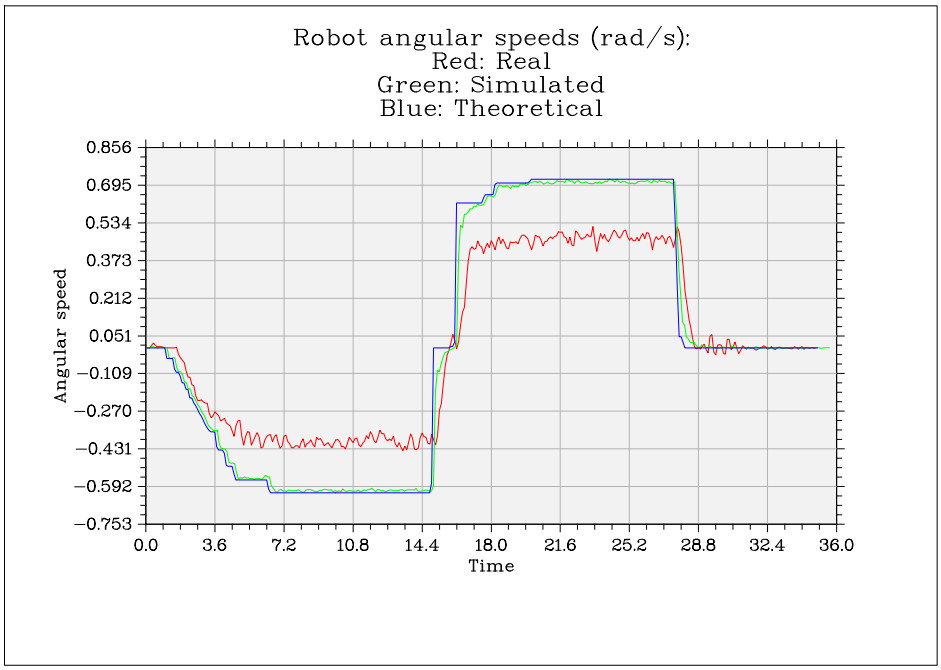


Figura E.27: Diferentes velocidades angulares de la prueba 7, ocho rápido antes del ajuste.

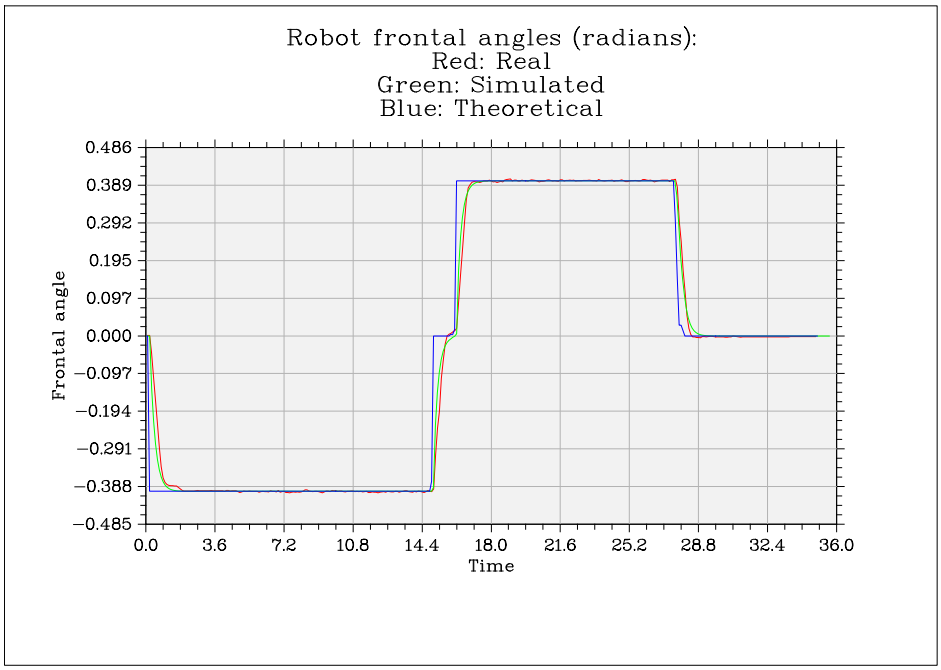


Figura E.28: Diferentes ángulos de la rueda virtual de la prueba 7, ocho rápido antes del ajuste.

Robot	Real	Simulado	Teórico
Distancia recorrida (m)	50,48	51,15	50,47
Tiempo total (s)	34,9	34,9	34,99
Velocidad media (m/s)	1,447	1,466	1,442
Vel. ang. media (rad/s)	0,2974	0,426	0,4396
Comparación	Sim./Real	Sim./Teór.	Real/Teór.
Separación final (m)	14,17	0,2932	14,46
Separación por metro	0,2771	0,005732	0,2865
Error vel. lineal (m/s)			
Media		-0,0458	
Desviación típica		0,0619	
Media abs.		0,0553	
Error vel. angular (rad/s)			
Media		-0,0098	
Desviación típica		0,1804	
Media abs.		0,1518	
Error ángulo (rad)			
Media		-0,0123	
Desviación típica		0,4034	
Media abs.		0,3444	
Correlación entre error vel. lineal y vel. lineal			
Pearson	0,3598	Valor p	1,539e-11
Spearman	0,2987	Valor p	3,809e-08
Correlación entre error vel. angular y vel. lineal			
Pearson	0,8924	Valor p	1,044e-120
Spearman	0,8864	Valor p	7,593e-117
Correlación entre error vel. lineal y vel. angular			
Pearson	0,3914	Valor p	1,276e-13
Spearman	0,2839	Valor p	1,935e-07
Correlación entre error vel. angular y vel. angular			
Pearson	0,9510	Valor p	1,395e-177
Spearman	0,9097	Valor p	3,606e-133

Tabla E.7: Resultados de la prueba 7, ocho rápido antes del ajuste.

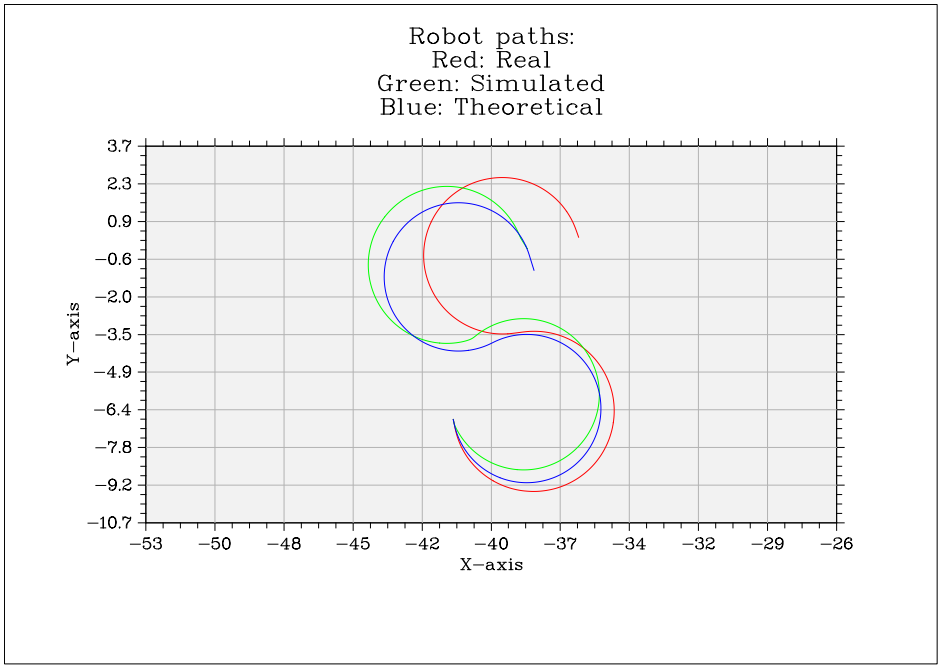


Figura E.29: Diferentes trayectorias de la prueba 8, ocho lento después del ajuste.

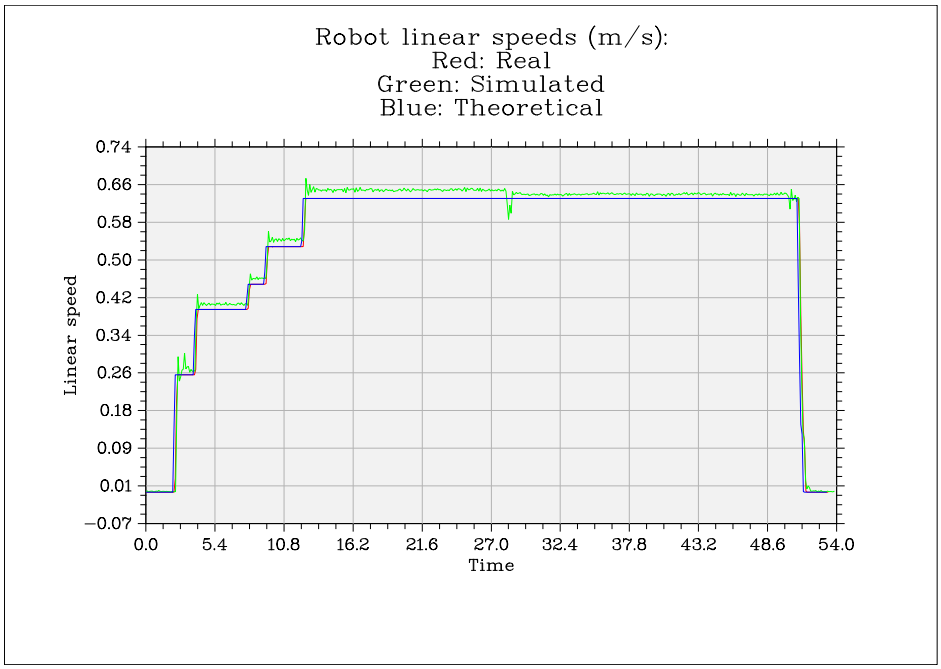


Figura E.30: Diferentes velocidades lineales de la prueba 8, ocho lento después del ajuste.

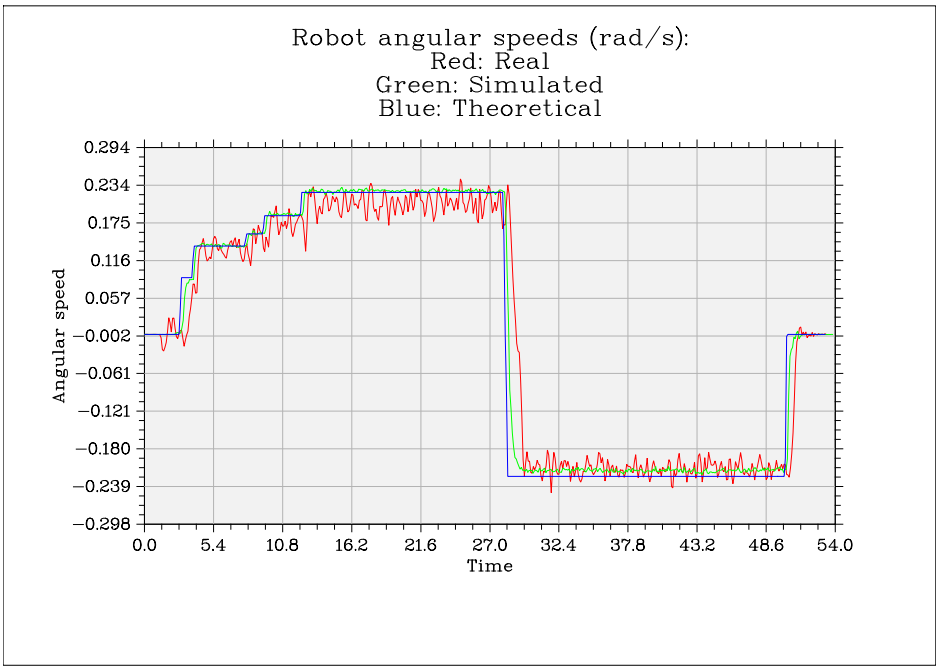


Figura E.31: Diferentes velocidades angulares de la prueba 8, ocho lento después del ajuste.

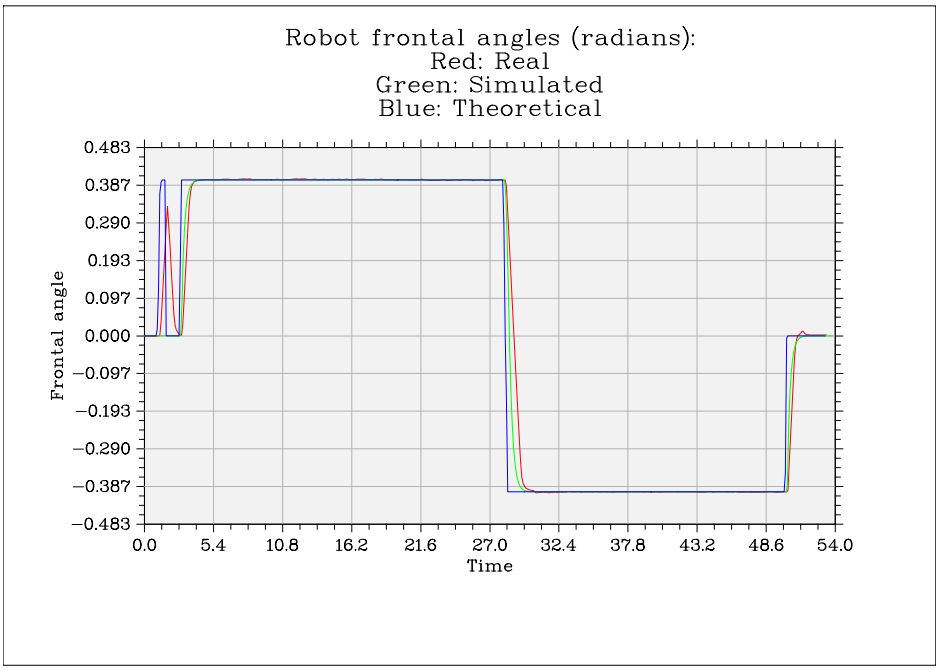


Figura E.32: Diferentes ángulos de la rueda virtual de la prueba 8, ocho lento después del ajuste.

Robot	Real	Simulado	Teórico
Distancia recorrida (m)	28,81	29,4	28,81
Tiempo total (s)	53,3	53,3	53,2
Velocidad media (m/s)	0,5406	0,5516	0,5416
Vel. ang. media (rad/s)	0,1722	0,1795	0,1849
Comparación	Sim./Real	Sim./Teór.	Real/Teór.
Separación final (m)	2,011	0,8574	2,126
Separación por metro	0,06839	0,02916	0,07379
Error vel. lineal (m/s)			
Media		-0,0108	
Desviación típica		0,0133	
Media abs.		0,0130	
Error vel. angular (rad/s)			
Media		-0,0043	
Desviación típica		0,0382	
Media abs.		0,0196	
Error ángulo (rad)			
Media		-0,0277	
Desviación típica		0,3710	
Media abs.		0,3488	
Correlación entre error vel. lineal y vel. lineal			
Pearson	0,1297	Valor p	0,004529
Spearman	0,6685	Valor p	5,046e-69
Correlación entre error vel. angular y vel. lineal			
Pearson	0,1363	Valor p	0,00283
Spearman	0,2821	Valor p	1,238e-10
Correlación entre error vel. lineal y vel. angular			
Pearson	0,0941	Valor p	0,03813
Spearman	0,5493	Valor p	2,658e-42
Correlación entre error vel. angular y vel. angular			
Pearson	-0,0584	Valor p	0,1619
Spearman	0,2656	Valor p	1,636e-09

Tabla E.8: Resultados de la prueba 8, ocho lento después del ajuste.

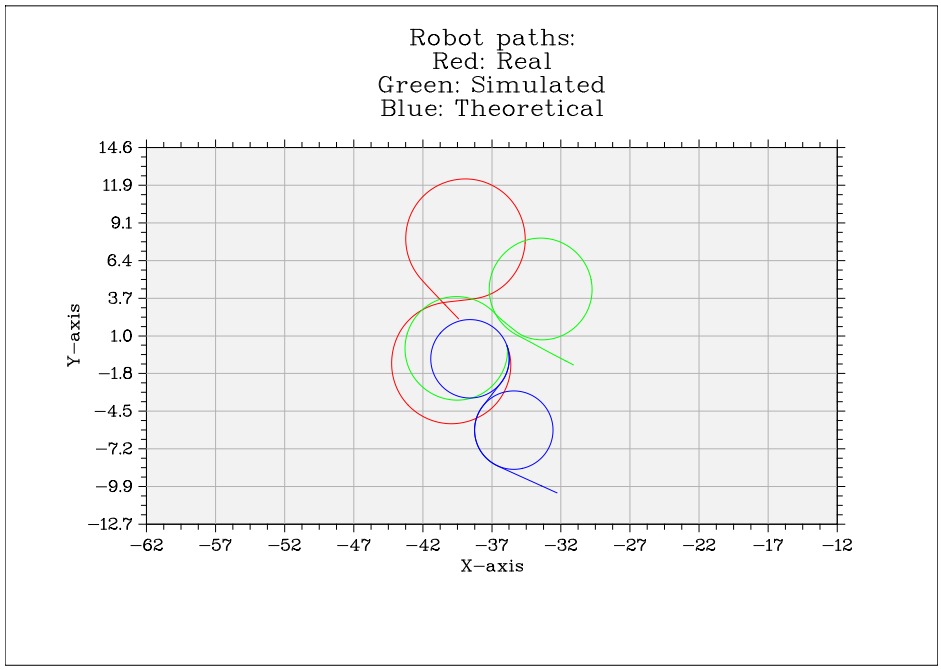


Figura E.33: Diferentes trayectorias de la prueba 9, ocho rápido después del ajuste.

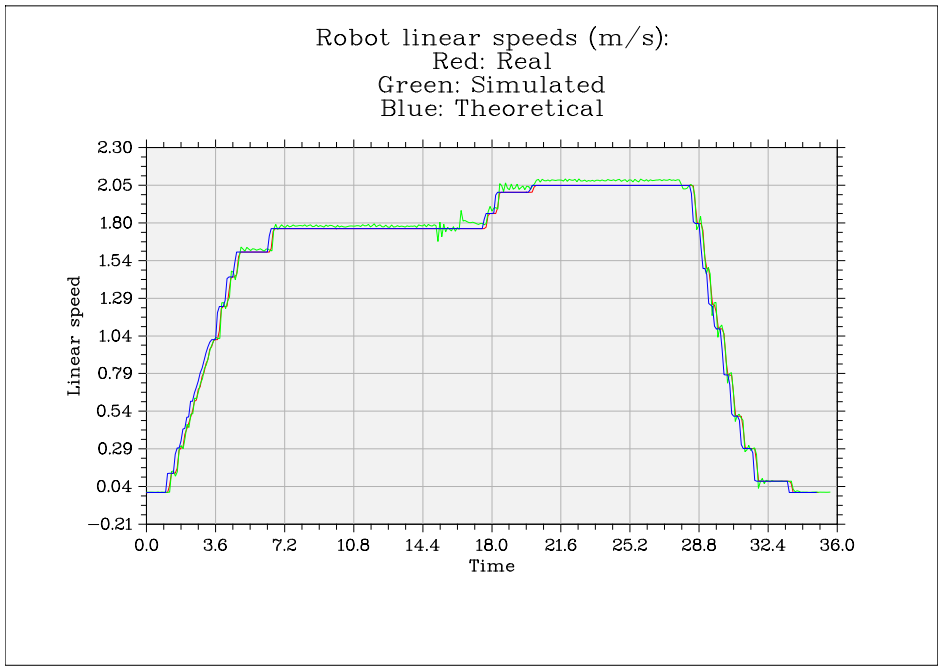


Figura E.34: Diferentes velocidades lineales de la prueba 9, ocho rápido después del ajuste.

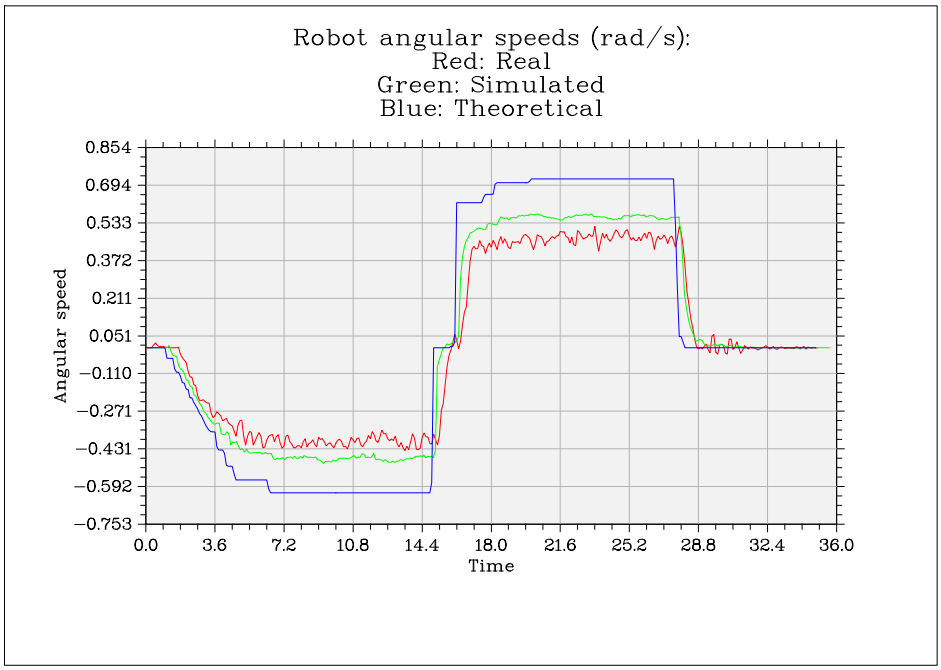


Figura E.35: Diferentes velocidades angulares de la prueba 9, ocho rápido después del ajuste.

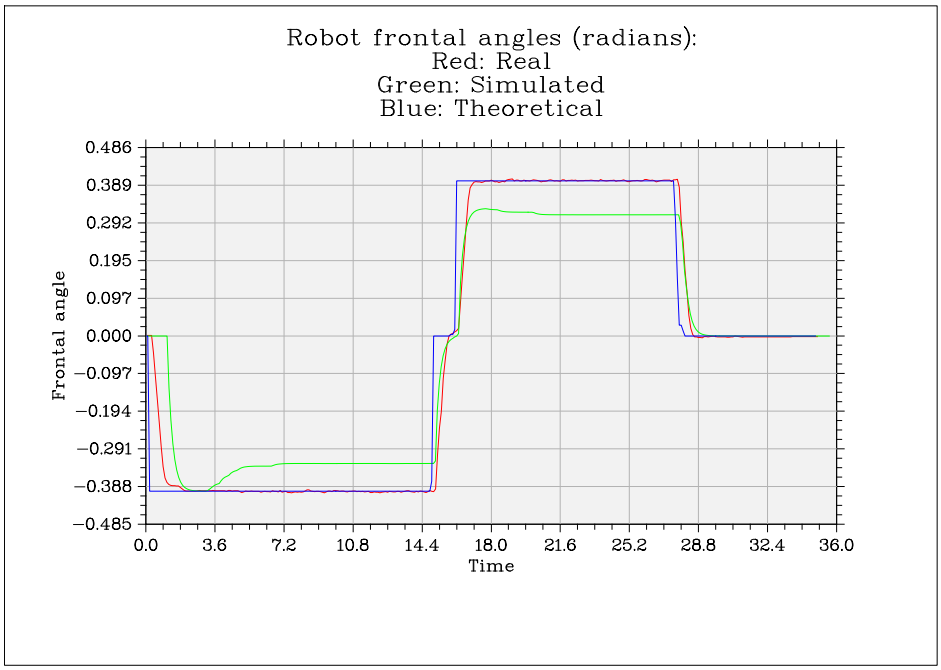


Figura E.36: Diferentes ángulos de la rueda virtual de la prueba 9, ocho rápido después del ajuste.

Robot	Real	Simulado	Teórico
Distancia recorrida (m)	50,48	50,99	50,47
Tiempo total (s)	35	35	34,9
Velocidad media (m/s)	1,442	1,457	1,446
Vel. ang. media (rad/s)	0,2965	0,342	0,4407
Comparación	Sim./Real	Sim./Teór.	Real/Teór.
Separación final (m)	8,955	9,339	14,47
Separación por metro	0,1756	0,1832	0,2865
Error vel. lineal (m/s)			
Media		-0,0114	
Desviación típica		0,0341	
Media abs.		0,0271	
Error vel. angular (rad/s)			
Media		-0,0103	
Desviación típica		0,0796	
Media abs.		0,0636	
Error ángulo (rad)			
Media		0,0184	
Desviación típica		0,2946	
Media abs.		0,2520	
Correlación entre error vel. lineal y vel. lineal			
Pearson	0,1809	Valor p	0,001242
Spearman	0,5069	Valor p	1,675e-23
Correlación entre error vel. angular y vel. lineal			
Pearson	0,6553	Valor p	2,14e-43
Spearman	0,7601	Valor p	4,216e-66
Correlación entre error vel. lineal y vel. angular			
Pearson	0,1520	Valor p	0,006907
Spearman	0,4302	Valor p	1,513e-16
Correlación entre error vel. angular y vel. angular			
Pearson	0,5581	Valor p	3,002e-29
Spearman	0,7886	Valor p	2,006e-74

Tabla E.9: Resultados de la prueba 9, ocho rápido después del ajuste.

E.4. Nuevas pruebas

Se realizan dos pruebas nuevas, llevadas a cabo en condiciones distintas a las del ajuste. La primera, prueba 10, es otra recta. La otra, prueba 11, es una trayectoria larga realizada a una velocidad reducida.

En la recta se observa que las trayectorias se ajustan bastante bien, aunque no a la perfección. El radio de las ruedas ha cambiado respecto a otras pruebas, lo que explica esa variación.

En la trayectoria larga los errores en la posición son mucho mayores. Sin embargo, en general las velocidades y los ángulos se acercan mucho, así que se puede considerar que principalmente son errores acumulados. La distancia total recorrida es relativamente grande, unos 130 metros. También hay que tener en cuenta que el ajuste se ha realizado en otras condiciones, algo que también introduce errores.

Las diferencias más significativas se producen en la velocidad angular. En primer lugar, se produce un pequeño deslizamiento en el robot real incluso para estas velocidades relativamente reducidas. Además, el robot simulado tiene unas ruedas con radios ligeramente distintos, lo que también introduce un poco de error.

Tanto el ajuste de la velocidad lineal como el de los ángulos es más preciso, las diferencias que existen son poco importantes.

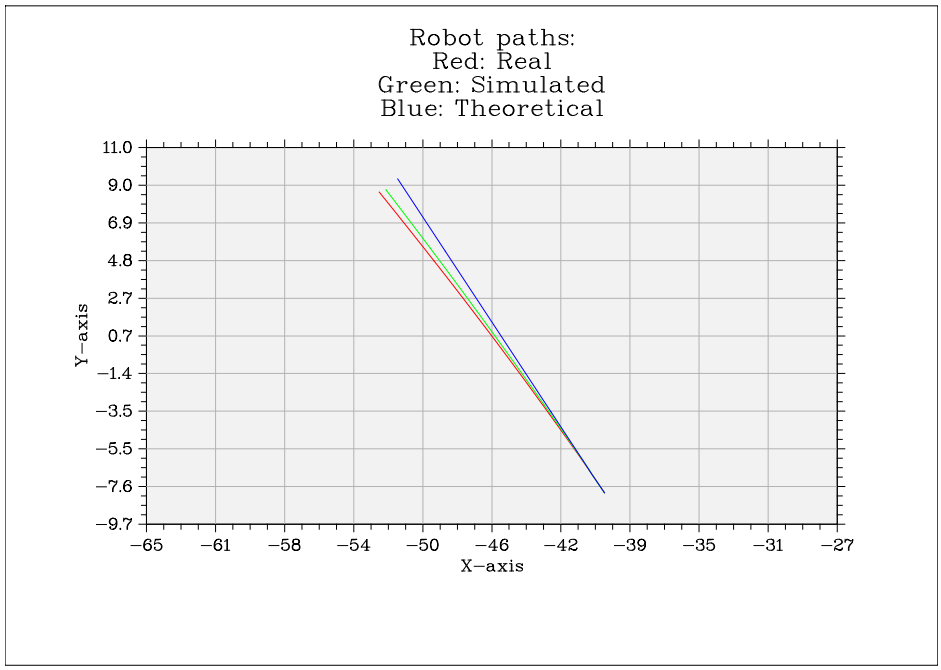


Figura E.37: Diferentes trayectorias de la prueba 10, nueva recta después del ajuste.

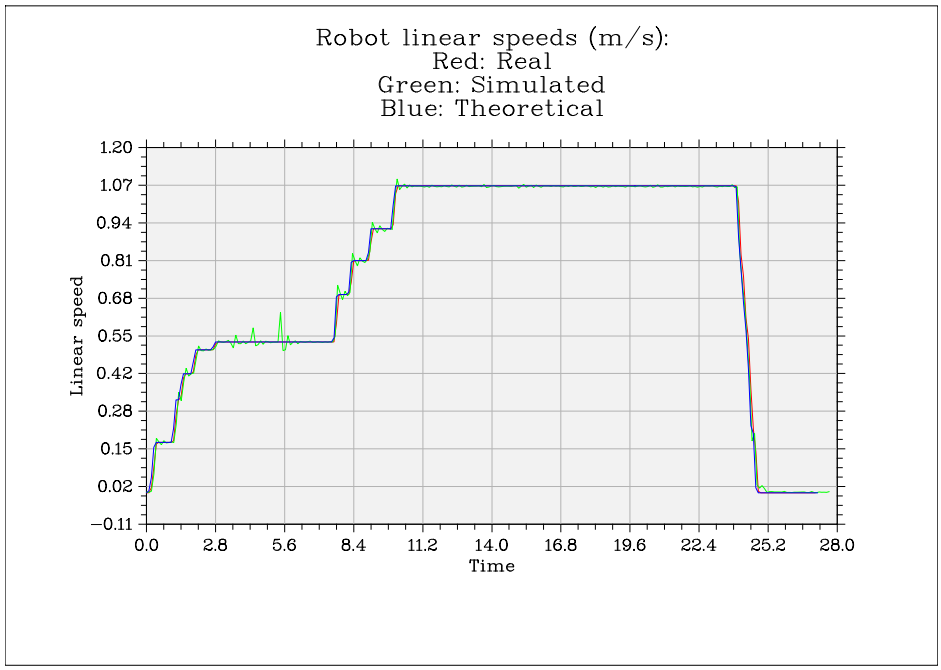


Figura E.38: Diferentes velocidades lineales de la prueba 10, nueva recta después del ajuste.

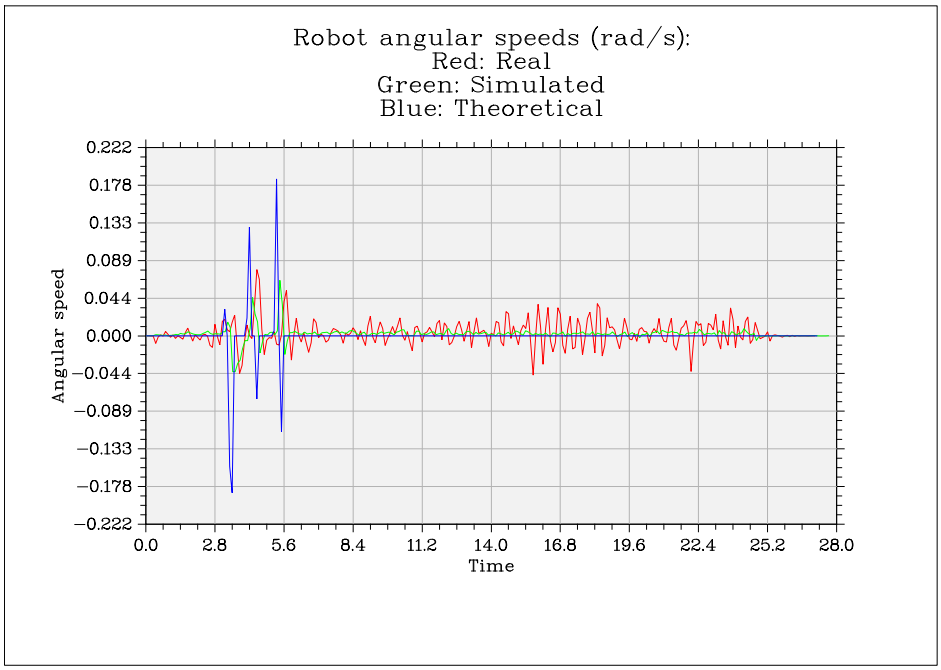


Figura E.39: Diferentes velocidades angulares de la prueba 10, nueva recta después del ajuste.

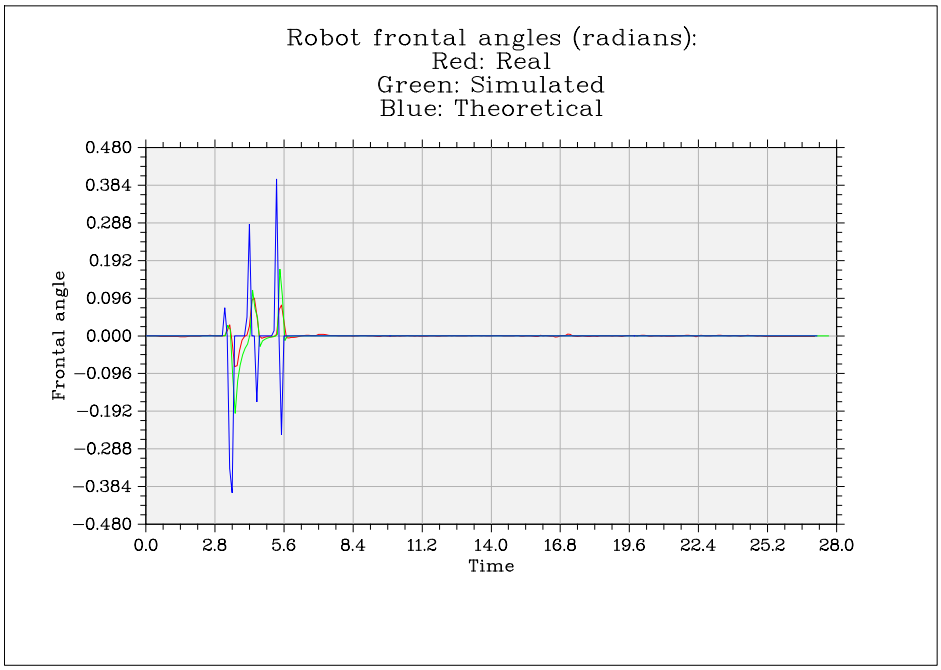


Figura E.40: Diferentes ángulos de la rueda virtual de la prueba 10, nueva recta después del ajuste.

Robot	Real	Simulado	Teórico
Distancia recorrida (m)	20,68	20,64	20,68
Tiempo total (s)	27,1	27,1	27,2
Velocidad media (m/s)	0,7632	0,7615	0,7605
Vel. ang. media (rad/s)	0,0105	0,004034	0,003302
Comparación	Sim./Real	Sim./Teór.	Real/Teór.
Separación final (m)	0,3876	0,8814	1,25
Separación por metro	0,01878	0,04271	0,06041
Error vel. lineal (m/s)			
Media		0,0019	
Desviación típica		0,0181	
Media abs.		0,0083	
Error vel. angular (rad/s)			
Media		0,0009	
Desviación típica		0,0153	
Media abs.		0,0105	
Error ángulo (rad)			
Media		0,0003	
Desviación típica		0,0220	
Media abs.		0,0046	
Correlación entre error vel. lineal y vel. lineal			
Pearson	-0,2779	Valor p	8,075e-06
Spearman	-0,3680	Valor p	1,258e-09
Correlación entre error vel. angular y vel. lineal			
Pearson	0,2665	Valor p	1,989e-05
Spearman	0,4150	Valor p	3,611e-12
Correlación entre error vel. lineal y vel. angular			
Pearson	0,3079	Valor p	6,05e-07
Spearman	0,1486	Valor p	0,01981
Correlación entre error vel. angular y vel. angular			
Pearson	0,4708	Valor p	9,26e-16
Spearman	0,3885	Valor p	1,099e-10

Tabla E.10: Resultados de la prueba 10, nueva recta después del ajuste.

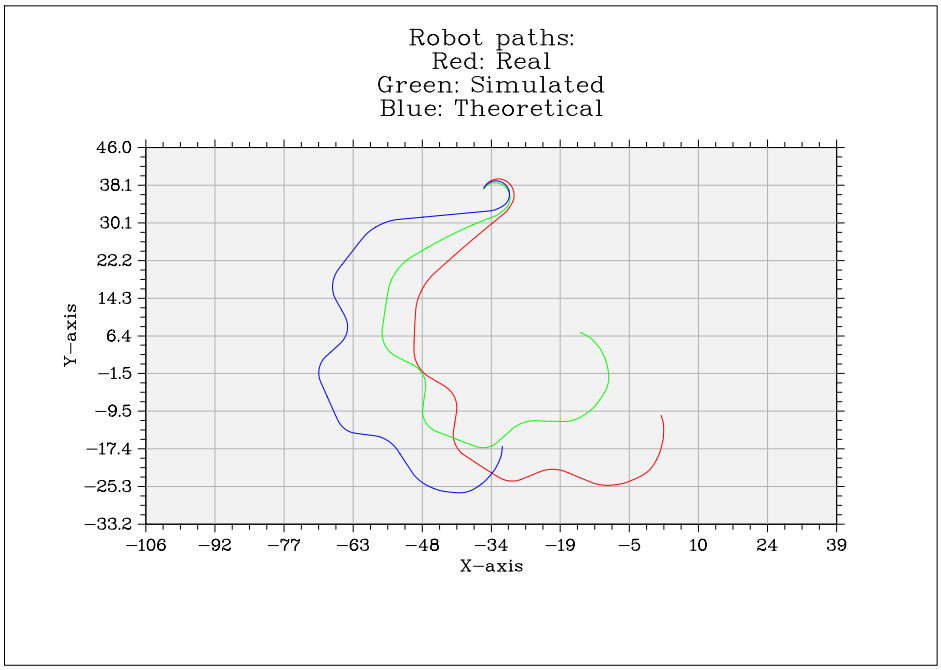


Figura E.41: Diferentes trayectorias de la prueba 11, nuevo camino después del ajuste.

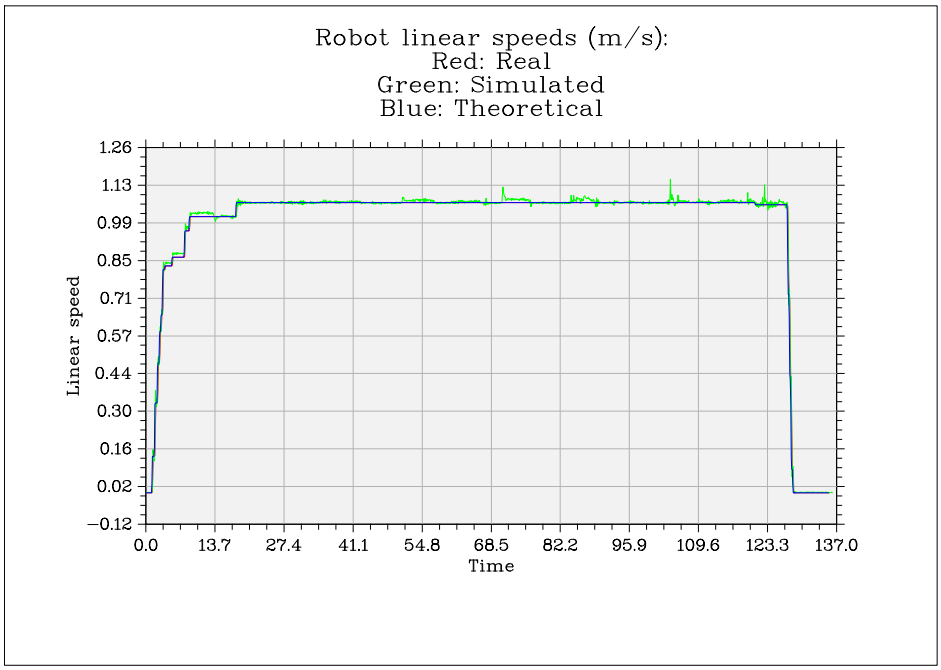


Figura E.42: Diferentes velocidades lineales de la prueba 11, nuevo camino después del ajuste.

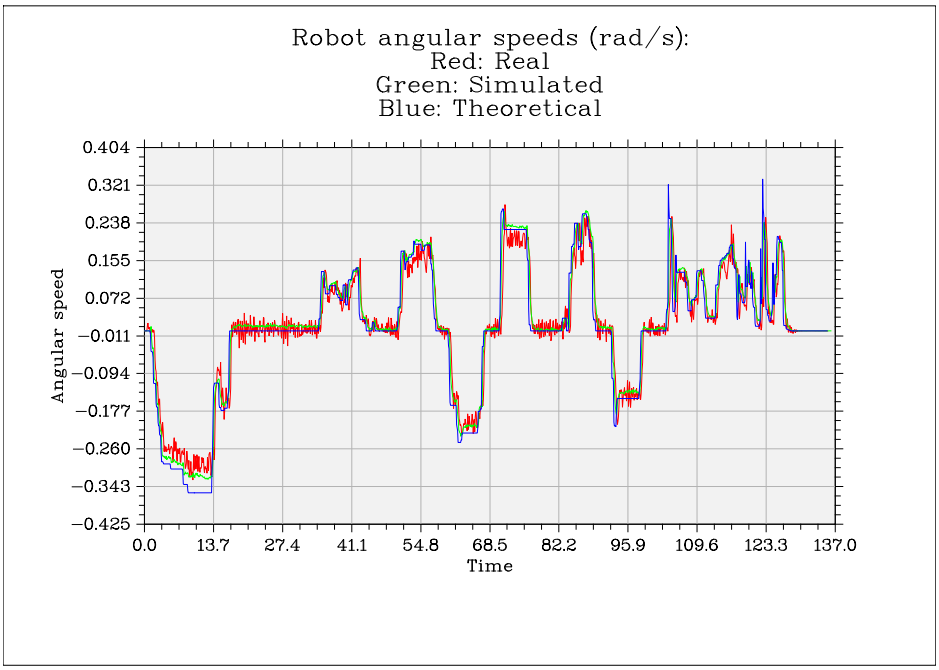


Figura E.43: Diferentes velocidades angulares de la prueba 11, nuevo camino después del ajuste.

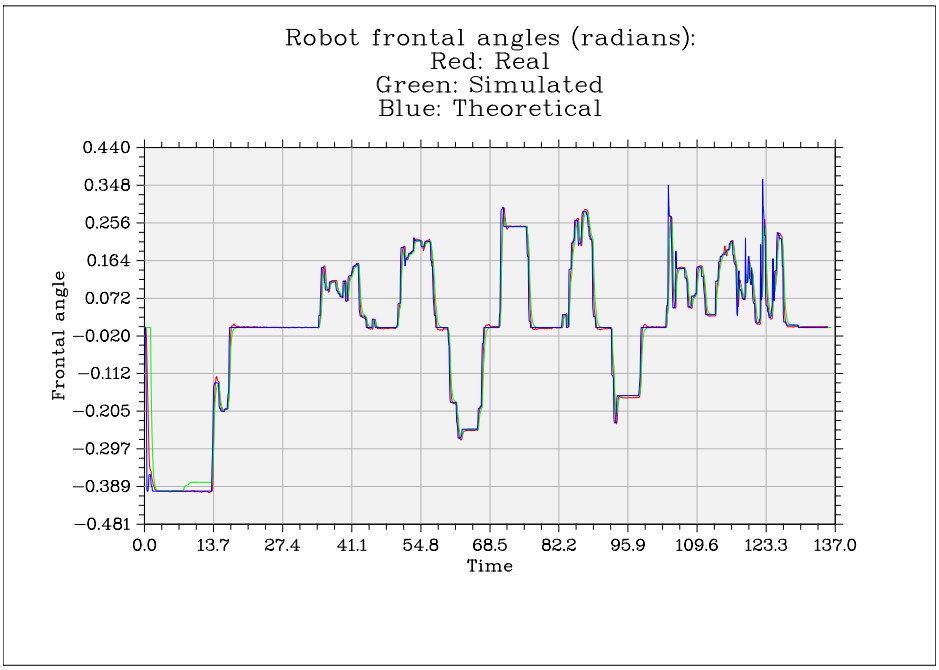


Figura E.44: Diferentes ángulos de la rueda virtual de la prueba 11, nuevo camino después del ajuste.

Robot	Real	Simulado	Teórico
Distancia recorrida (m)	131,4	131,9	131,4
Tiempo total (s)	135,5	135,5	135,4
Velocidad media (m/s)	0,9701	0,9734	0,9708
Vel. ang. media (rad/s)	0,09009	0,09777	0,09665
Comparación	Sim./Real	Sim./Teór.	Real/Teór.
Separación final (m)	24,33	29,08	34
Separación por metro	0,1845	0,2204	0,2587
Error vel. lineal (m/s)			
Media		-0,0032	
Desviación típica		0,0092	
Media abs.		0,0059	
Error vel. angular (rad/s)			
Media		-0,0054	
Desviación típica		0,0258	
Media abs.		0,0180	
Error ángulo (rad)			
Media		0,0010	
Desviación típica		0,1671	
Media abs.		0,1139	
Correlación entre error vel. lineal y vel. lineal			
Pearson	-0,0385	Valor p	0,1478
Spearman	0,1989	Valor p	6,837e-13
Correlación entre error vel. angular y vel. lineal			
Pearson	0,2031	Valor p	2,111e-13
Spearman	0,2346	Valor p	1,258e-17
Correlación entre error vel. lineal y vel. angular			
Pearson	0,4054	Valor p	1,882e-53
Spearman	0,5751	Valor p	3,757e-118
Correlación entre error vel. angular y vel. angular			
Pearson	0,3399	Valor p	6,549e-37
Spearman	0,4554	Valor p	7,402e-69

Tabla E.11: Resultados de la prueba 11, nuevo camino después del ajuste.

E.5. Conclusiones

El error que se producía en las rectas debido a los radios de las ruedas prácticamente se ha eliminado de la recta con la que se ajustó. La separación final se ha reducido aproximadamente 36 veces. Al probar con otra, se puede observar que se aproxima bastante, pero sí existe un pequeño error. Esto se debe a que la otra recta se probó en condiciones distintas, y cualquier pequeña variación en los tamaños de las ruedas se traduce en errores considerables. De cualquier forma, este error es más de diez veces más pequeño que antes del ajuste, de unos tres milímetros por cada metro recorrido.

Para una trayectoria con curvas a una velocidad elevada los resultados se acercan más que antes de ajustar, reduciéndose el error en un 228%. Sin embargo, siguen sin ser muy precisos, con un error medio de 0,0043 rad/s. Esto se debe a que la corrección de velocidad angular se realiza para unas condiciones en concreto, que no son las de esta trayectoria. Sería necesario reajustar para esta situación o simplemente asumir que para altas velocidades angulares existe un error que debe ser tenido en cuenta.

Al probar con una trayectoria más larga, en la prueba 11, se observa que los errores de posición, acumulados a lo largo del ensayo, son bastante considerables. Sin embargo, las velocidades lineales y los ángulos presentan errores muy reducidos. La principal causa del error en este caso es, de nuevo, la velocidad angular, con una media de 0,0054 rad/s.

En general, se puede decir que el simulador es bastante realista para velocidades angulares reducidas. En cuanto el fenómeno del deslizamiento empieza a ser relevante los resultados empeoran considerablemente. Las posiciones terminan teniendo errores grandes, pero se debe principalmente a que se acumulan. Las velocidades lineales y los ángulos de la rueda virtual central se reproducen con exactitud, al igual que la velpcidad angular si ésta es reducida.

ANEXO F

CONFIGURACIÓN DEL SIMULADOR

Este anexo trata sobre los parámetros del simulador que se pueden controlar modificando el fichero *configuration.urdf.xacro* de la descripción URDF del robot.

Todos los parámetros se introducen con el siguiente formato:

```
<xacro:property name="nombre" value="valor" />
```

El campo *name* indica su nombre y el campo *value* el valor que toma. Xacro es un lenguaje de macros de XML que permite simplificar las descripciones de los robots. En este caso se utiliza para poder modificar todos los valores al principio del fichero. Las macros se encargan después de colocar esos valores en sus lugares correspondientes.

El simulador permite utilizar un valor distinto para los cálculos que utilizan el radio de las ruedas y para cada rueda del modelo. *wheel_radius* es el teórico usado para esos cálculos, mientras que los otros cuatro son los del modelo. Todos los valores deben estar en metros.

```
name="wheel_radius" value="0.3"
name="front_left_radius" value="0.294"
name="front_right_radius" value="0.299"
name="back_left_radius" value="0.300"
name="back_right_radius" value="0.307"
```

La velocidad lineal y el ángulo virtual central máximos se controlan limitando las órdenes recibidas. Limitar directamente los ángulos de las ruedas provoca comportamientos extraños cuando éstas se acercan a ese límite. El parámetro que limita la velocidad lineal es *wheel_turn_software_limit*, en metros por segundo. *robot_linear_speed_limit* indica el valor máximo que puede tomar el ángulo, expresado en radianes.

```
name="wheel_turn_software_limit" value="0.4"
name="robot_linear_speed_limit" value="4.0"
```

La frecuencia con la que se actualiza el control de las ruedas, tanto el de la velocidad de giro como el del PID de los ángulos, se fija con *control_update_rate*. Su valor indica el período de actualización, en segundos.

```
name="control_update_rate" value="0.1"
```

Las constantes del controlador PID del ángulo de las ruedas se controlan con los siguientes parámetros. El término proporcional es *pid_kp*, el integral es *pid_ki* y el derivativo *pid_kd*. Hay que destacar que pequeños cambios en estos valores pueden provocar que el control pase a ser inestable. Por lo tanto, se recomienda modificarlos con cuidado.

```
name="pid_kp" value="3.5"
name="pid_ki" value="0.01"
name="pid_kd" value="0.02"
```

Para activar la corrección del ángulo de las ruedas para simular el deslizamiento hay que introducir un valor no nulo en *sliding_on*. El valor de velocidas angular a partir del cual empieza a tener efecto es *lower_limit*. A partir de ahí aumenta linealmente con la orden de velocidad angular hasta que ésta llega a *upper_limit_theo*, punto en el que la del simulador será *upper_limit_real*. A partir de ahí se mantiene constante. Todas esas velocidades angulares deben expresarse en radianes por segundo.

```
name="sliding_on" value="1.0"
name="lower_limit" value="0.3"
name="upper_limit_real" value="0.55"
name="upper_limit_theo" value="0.7"
```

Es importante tener en cuenta que utilizar esta correción provoca que el ángulo deje de ser correcto, buscando una velocidad angular más realista para el caso para el que se hayan calculado los parámetros.

Finalmente, las varianzas del desplazamiento del robot y del ángulo de la rueda central virtual son respectivamente *sigma_s* y *sigma_phi*. Estos valores se usan después para el cómputo de la matriz de covarianzas, como se explica en la sección 4.3.1.

```
name="sigma_s^2" value="0.0"
name="sigma_phi^2" value="0.0"
name="sigma_s_phi" value="0.0"
```

ANEXO G

CÁLCULO DE LA MATRIZ DE COVARIANZAS

En este anexo se describe con más detalle cómo se calcula la matriz de covarianzas utilizada para la estimación del error de la sección 4.3.1 en el capítulo 4. Σ es esa matriz de covarianzas:

$$\Sigma = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 \end{pmatrix}, \tag{G.1}$$

donde los elementos de la diagonal principal son las varianzas de los errores en las coordenadas cartesianas x e y y la orientación θ. El resto de elementos son sus covarianzas.

Se necesita obtener la relación con las varianzas de la distancia recorrida, s, y el ángulo de la rueda virtual central, ϕ. Se utiliza un desarrollo de Taylor de primer orden para aproximar esa relación:

$$\Sigma = J\,Q\,J^T. \tag{G.2}$$

La matriz Q contiene esas dos varianzas, además de la covarianza. El error de s aumenta linealmente con la velocidad y el de ϕ se considera constante.

$$Q = \begin{pmatrix} |s| \cdot \sigma_s^2 & \sigma_{s\phi} \\ \sigma_{s\phi} & \sigma_\phi^2 \end{pmatrix}. \tag{G.3}$$

J es la matriz jacobiana de la función que relaciona los incrementos de posición y ángulo con s y ϕ:

$$J = \begin{pmatrix} \frac{\partial \Delta x}{\partial s} & \frac{\partial \Delta x}{\partial \phi} \\ \frac{\partial \Delta y}{\partial s} & \frac{\partial \Delta y}{\partial \phi} \\ \frac{\partial \Delta \theta}{\partial s} & \frac{\partial \Delta \theta}{\partial \phi} \end{pmatrix}. \tag{G.4}$$

Para obtener esta matriz se comienza con las variaciones de la posición y la orientación, para un robot sin deslizamiento:

$$\dot{x} = v\cos\theta, \tag{G.5}$$

$$\dot{y} = v\,\mathrm{sen}\,\theta, \tag{G.6}$$

$$\dot{\theta} = \frac{v\tan\phi}{L}, \tag{G.7}$$

donde v es la velocidad lineal y L la distancia entre ejes.

Integrando respecto al tiempo se llega a:

$$\Delta x = vt\cos\theta, \tag{G.8}$$

$$\Delta y = vt\,\mathrm{sen}\,\theta, \tag{G.9}$$

$$\Delta\theta = \frac{vt\tan\phi}{L}. \tag{G.10}$$

Teniendo en cuenta que la distancia recorrida es igual a la velocidad lineal por el tiempo, estas ecuaciones también se pueden expresar en función de la distancia recorrida s:

$$\Delta x = s\cos\theta, \tag{G.11}$$

$$\Delta y = s\,\mathrm{sen}\,\theta, \tag{G.12}$$

$$\Delta\theta = \frac{s\tan\phi}{L}. \tag{G.13}$$

Derivando respecto a s se obtienen los elementos de la primera columna de la matriz jacobiana:

$$\frac{\partial x}{\partial s} = \cos\theta, \tag{G.14}$$

$$\frac{\partial y}{\partial s} = \mathrm{sen}\,\theta, \tag{G.15}$$

$$\frac{\partial \theta}{\partial s} = \frac{\tan\phi}{L}. \tag{G.16}$$

Y los otros tres se obtienen derivando respecto a ϕ:

$$\frac{\partial x}{\partial \phi} = \begin{cases} \dfrac{(1+\tan^2\phi)\left(s\cdot\cos(\Delta\theta) - \dfrac{L\cdot\operatorname{sen}(\Delta\theta)}{\tan\phi}\right)}{\tan\phi} & \text{si } \Delta\theta \neq 0 \\ -s\cdot\operatorname{sen}\theta\,\dfrac{\tan\phi}{L} & \text{si } \Delta\theta = 0, \end{cases} \tag{G.17}$$

$$\frac{\partial y}{\partial \phi} = \begin{cases} \dfrac{(1+\tan^2\phi)\left(s\cdot\operatorname{sen}(\Delta\theta) - \dfrac{L\cdot(\cos(\Delta\theta)-1)}{\tan\phi}\right)}{\tan\phi} & \text{si } \Delta\theta \neq 0 \\ s\cdot\cos\theta\,\dfrac{\tan\phi}{L} & \text{si } \Delta\theta = 0, \end{cases} \tag{G.18}$$

$$\frac{\partial \theta}{\partial \phi} = \frac{s(1+\tan^2\theta)}{L}. \tag{G.19}$$

El desplazamiento lineal s se calcula utilizando:

$$s = \begin{cases} \Delta\theta\,\dfrac{\Delta x^2 + \Delta y^2}{2\Delta y} & \text{si } \Delta\theta \neq 0 \\ \Delta x & \text{si } \Delta\theta = 0. \end{cases} \tag{G.20}$$

Para el ángulo ϕ se usa:

$$\phi = \tan^{-1}\left(\frac{2\cdot L\cdot\Delta y}{\Delta x^2 + \Delta y^2}\right). \tag{G.21}$$

ANEXO H

ANIMACIONES DEL SIMULADOR

Se añade al proyecto un CD-ROM que incluye algunas animaciones del robot en el simulador:

- Robot moviéndose por un escenario en modo coche.
- Ejemplo del movimiento del robot en modo dual.
- Ejemplo del movimiento del robot en modo *crab*.

También se adjunta una carpeta con los vídeos correspondientes a las pruebas del anexo E, etiquetadas con la numeración de ese anexo.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

BIBLIOGRAFÍA

[1] Gazebo, "Simulador multi-robot 3D." `http://gazebosim.org/`.

[2] ROS, "Robot Operating System." `http://www.ros.org`.

[3] pr2_simulator, "Simulador para Gazebo del robot PR2." `http://www.ros.org/wiki/pr2_simulator`.

[4] turtlebot_simulator, "Simulador para Gazebo del robot TurtleBot." `http://ros.org/wiki/turtlebot_simulator`.

[5] Kinect, "Kinect para Xbox 360." `http://www.xbox.com/es-ES/Kinect/GetStarted`.

[6] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, y A. Fitzgibbon, "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera," en *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, (New York, NY, USA), pp. 559–568, ACM, 2011.

[7] P. Henry, M. Krainin, E. Herbst, X. Ren, y D. Fox, "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[8] A. King, "Inertial navigation - forty years of evolution," *GEC review*, vol. 13, no. 3, pp. 140–149, 1998.

[9] M. Barquins y A. D. Roberts, "Rubber friction variation with rate and temperature: some new observations," *Journal of Physics D: Applied Physics*, vol. 19, no. 4, p. 547, 1986.

[10] R. Pinnington, "Rubber friction on rough and smooth surfaces," *Wear*, vol. 267, pp. 1653 – 1664, 2009.

[11] fuerte, "ROS Fuerte Turtle." http://ros.org/wiki/fuerte.

[12] diamondback, "ROS Diamondback." http://ros.org/wiki/diamondback.

[13] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, y A. Y. Ng, "ROS: an open-source Robot Operating System," en *ICRA Workshop on Open Source Software*, 2009.

[14] J. C. Trinkle, J.-S. Pang, S. Sudarsky, y G. Lo, "On dynamic multi-rigid-body contact problems with coulomb friction," *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 77, no. 4, pp. 267–279, 1997.

[15] N. Koenig y A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," en *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, pp. 2149–2154, IEEE.

[16] Penn Robotics, "Matlab-ROS IPC Bridge." https://alliance.seas.upenn.edu/~meam620/wiki/index.php?n=Roslab.IpcBridge.

[17] W. Mendenhall y T. Sincich, *Probabilidad y estadística para ingeniería y ciencias*. Prentice Hall Hispanoamericana, 4ª ed., 1997.

[18] G. Bishop y G. Welch, "An introduction to the Kalman filter," *Proc of SIGGRAPH, Course*, vol. 8, pp. 27599–3175, 2001.

[19] A. Kelly, "Linearized error propagation in odometry," *The International Journal of Robotics Research*, vol. 23, no. 2, pp. 179–218, 2004.

[20] B. Freedman, A. Shpunt, M. Machline, y Y. Arieli, "Depth mapping using projected patterns," Abr. 2 2008. US Patent App. 12/522,171.

[21] K. Khoshelham y S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[22] PCL, "Point Cloud Library." http://pointclouds.org/.

[23] R. B. Rusu y S. Cousins, "3D is here: Point Cloud Library (PCL)," en *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), May 9-13 2011.

[24] OpenCV, "Open Source Computer Vision." `http://opencv.org/`.

[25] H. Bay, T. Tuytelaars, y L. Van Gool, "Surf: Speeded up robust features," *Computer Vision–ECCV 2006*, pp. 404–417, 2006.

[26] M. Muja y D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," en *International Conference on Computer Vision Theory and Applications (VISSAPP'09)*, pp. 331–340, 2009.

[27] M. A. Fischler y R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[28] S. Rusinkiewicz y M. Levoy, "Efficient variants of the ICP algorithm," en *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pp. 145–152, IEEE, 2001.

[29] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, y M. Beetz, "Towards 3D point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927–941, 2008.

[30] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, y C. T. Silva, "Computing and rendering point set surfaces," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 9, no. 1, pp. 3–15, 2003.

[31] C. T. Silva, J. S. Mitchell, y A. E. Kaufman, "Automatic generation of triangular irregular networks using greedy cuts," en *Proceedings of the 6th conference on Visualization'95*, p. 201, IEEE Computer Society, 1995.

[32] M. Kazhdan, M. Bolitho, y H. Hoppe, "Poisson surface reconstruction," en *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006.

[33] R. Li, L. Liu, L. Phan, S. Abeysinghe, C. Grimm, y T. Ju, "Polygonizing extremal surfaces with manifold guarantees," en *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, pp. 189–194, ACM, 2010.

[34] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, y H.-P. Seidel, "Laplacian surface editing," en *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pp. 175–184, ACM, 2004.

[35] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," en *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 1403–1410, IEEE, 2003.

[36] F. Lu y E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous robots*, vol. 4, no. 4, pp. 333–349, 1997.

[37] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, y J. Hertzberg, "The efficient extension of globally consistent scan matching to 6 dof," *Knowledge Based Systems*, vol. 1, p. 20, 2008.

[38] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, y A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," en *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pp. 127–136, IEEE, 2011.

[39] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, y J. McDonald, "Kintinuous: Spatially extended KinectFusion," en *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, (Sydney, Australia), Jul 2012.

[40] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, y W. Stuetzle, *Surface reconstruction from unorganized points*, vol. 26. ACM, 1992.

[41] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, y T. R. Evans, "Reconstruction and representation of 3d objects with radial basis functions," en *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 67–76, ACM, 2001.

[42] gazebo_plugins, "Plugins de sensores para Gazebo." `http://ros.org/wiki/gazebo_plugins`.

[43] hector_gazebo, "Plugins de Team Hector para Gazebo." `http://www.ros.org/wiki/hector_gazebo`.

[44] gps_common, "GPS Common." `http://www.ros.org/wiki/gps_common`.